UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

CLASIFICACIÓN DE EVENTOS EN SEÑALES TEMPORALES PROCEDENTES DE SENSORES INDUSTRIALES

Autor: Labrador Serrano, Beltrán
Tutor: Ramos Castro, Daniel

JUNIO 2017

Clasificación de Eventos en Señales Temporales Procedentes de Sensores Industriales

Autor: Labrador Serrano, Beltrán
Tutor: Ramos Castro, Daniel

Grupo Audias-ATVS
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2017

Resumen

Este Trabajo Fin de Grado consiste en el desarrollo de diferentes clasificadores de eventos de señales provenientes de la inspección de piezas con sensores industriales, con el fin de detectar irregularidades y errores producidos durante el proceso de fabricación, un control de calidad necesario al final de la cadena de producción.

Con este fin, se ha propuesto en este trabajo, el desarrollo de un clasificador de eventos dentro de un sistema completo de reconocimiento de patrones. Este sistema completo presenta un esquema *Front-end – Back-end*, estando este trabajo centrado en el *Back-end*, que clasifica los eventos a partir de ciertas puntuaciones obtenidas por el *Front-end*. A su vez también se afronta en este TFG la tarea de transformar los datos para su mejor clasificación y la posterior medida de rendimiento del sistema completo.

El *back-end* se ha desarrollado usando funciones de densidad de probabilidad con distribución gaussiana multivariada, estimándose mediante máxima verosimilitud (ML) y mediante modelados bayesianos, esto último con el fin de tener un clasificador robusto en un escenario de pocos datos de entrenamiento, como puede ser el caso real en la inspección de piezas industriales. Se comprueba que cuando existe una gran cantidad de datos de entrenamiento, los dos modelos son equivalentes, sin embargo, en un escenario de poca escasez de datos, el modelado bayesiano de la distribución gaussiana supera en rendimiento a la estimación mediante Máxima Verosimilitud.

Además, se ha implementado un algoritmo de *feature warping*, con el fin de *gaussianizar* los datos de entrada al back-end, pudiéndose este aislar de los posibles cambios en el Front-end, y permitir un mejor modelado de los datos mediante una distribución gaussiana.

Por último, se ha implementado una fusión de diferentes detectores en el Back-end con el fin de inmunizar el clasificador a los fallos de los detectores individuales, mientras estos errores no estén correlados, y mejorar la clasificación.

Abstract

This Bachelor Thesis consists on the development of several event classifiers working on signals resulting from the inspection of parts with industrial sensors, with the final purpose of detecting irregularities and errors produced during the manufacturing process. This quality control, placed at the end of the production chain, is of utmost importance to be able to assure that the final product meets the quality standards.

In order of achieving this goal, it has been proposed in this Bachelor Thesis, the development of an event classifier within a full pattern recognition system. This system is structured with a Front-end – Back-end scheme, being this work centered on the Back-end, classifying the events using a series of scores obtained by the Front-end. This work also faces the task of transforming the data for its better classification, and the performance measures of the full system also.

The back-end has been developed using probability density functions of multivariate Gaussian distributions, estimated by means of Maximum Likelihood optimization (ML) and by means of Bayesian models, in order to have a more robust classifier in a low training data environment, as could be the real case in the industrial parts inspection business. It is concluded that when there is a large number of training samples, both models are equivalent, but on the contrary, when there is training data shortage, the Bayesian modeling of the Gaussian distribution has a better performance than the Maximum Likelihood estimation.

Furthermore, a Feature Warping algorithm has been implemented in order to transform the input data to the Back-end to a more gaussian distribution, being able to isolate the back-end from possible changes on the Front-end, and allowing to do a better modeling of the samples with a gaussian distribution.

Finally, a score fusion from different detectors has been implemented, to be able to immunize the classifier from the errors of the individual detectors, as long as these errors are not correlated, and improve the classification.

Palabras clave

Clasificación de eventos

Back-end

Front-end

Feature Warping

Modelado bayesiano

Distribución gaussiana multivariada

Fusión de detectores

Keywords

Event classification

Back-end

Front-end

Feature Warping

Bayesian modeling

Multivariate gaussian distribution

Score fusion

Agradecimientos

Con este Trabajo de Fin de Grado acaba una etapa de mi vida muy importante. Aunque aún quedan unos años de universidad, este punto es el final de un ciclo y el comienzo de otro.

Para mí entrar en la universidad fue un cambio de envergadura, después de un bachillerato estudiando a distancia, en el internado de Francia, después del año en EEUU… La universidad ha sido para mí el primer periodo de mi vida en el que he estado en un único lugar, donde he podido afianzar relaciones personales con otra gente a lo largo del tiempo.

Y es el tiempo el que quiero agradecer a todos vosotros, los que habéis estado conmigo estos años. Ese tiempo que me habéis dado y ese tiempo que he compartido con vosotros.

Agradecértelo todo a ti, Belén, por cambiar mi forma de ver el mundo, por enseñarme a soñar y a no querer dormir.

Agradeceros a vosotros, Álvaro Iglesias, Carlos Lamas y Alex Martín, mis compañeros de la universidad por excelencia. Por sacarme de apuros mil veces, por todo el tiempo que hemos pasado juntos, en la universidad y fuera. Sin vosotros no habría sido lo mismo.

A Pablo Vicente, con el que tanto he compartido estos años.

A toda la gente de Miraflores, por enseñarme que la vida no es sólo estudiar.

A Pablo Piñeiro, por todo lo que me ha enseñado.

A mi familia, mis padres y mi hermano Rodrigo, que siempre estaréis conmigo.

A mi tutor, Daniel Ramos, por permitirme hacer este TFG, por confiar en mí y por abrirme las puertas de Audias.

Y finalmente a todo el mundo que me ha ayudado a llegar hasta aquí, a todos los profesores que me han motivado, de la universidad y de antes.

Gracias.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Este trabajo pretende transferir técnicas sofisticadas usadas en audio en problemas complejos, como el reconocimiento de idioma, a la inspección de piezas industriales, con el objetivo de detectar irregularidades que se manifiestan como eventos en las señales adquiridas por ciertos sensores.

Esta detección de irregularidades, realizada al final de la cadena de producción, es de máxima importancia en el control de calidad del producto final para certificar que éste cumple con los estándares. El uso de técnicas de *machine learning* y reconocimiento de patrones amplía las posibilidades de detección y permite la detección automática de los errores de producción.

El análisis de estas señales entrenamiento de estos algoritmos de detección automática es crítico, debido a la posible insuficiencia de datos de entrenamiento en el entorno de trabajo. En este escenario, es importante desarrollar algoritmos de mayor robustez a la escasez de datos de entrenamiento, tanto de forma general como de un tipo de evento en concreto.

Al pasar la pieza en movimiento a inspeccionar por el sensor, este genera una señal que responde a las propiedades internas de la pieza. Estas son las señales que se analizan en este TFG, y los eventos en esas señales los que se pretenden clasificar.

Al pasar una misma pieza por el sensor varias veces, genera señales distintas debido al ruido de fondo. Parte del objetivo del sistema completo es conseguir obtener la información de la señal extrayendo cuidadosamente alguna de sus características, obviando la información del ruido.

1.2 Objetivos

Con esta motivación presente se han marcado los siguientes objetivos:

1. Realizar un análisis estadístico de los scores de los detectores de entrada con el fin de realizar un mejor modelado de los datos.

2. Desarrollar un clasificador que tome como entrada las puntuaciones de un conjunto de detectores y que identifique y clasifique los eventos de un conjunto de entrenamiento mediante *cross validation*.

3. Implementar un clasificador que sea robusto en un escenario con pocos datos de entrenamiento de todos o escasez de algún tipo de evento en concreto.

4. Transformar los datos de entrada del clasificador para ajustarlos mejor a la distribución objetivo y realizar un mejor modelado con la distribución disponible.

5. Realizar un análisis de rendimiento del sistema completo con el fin de analizar las limitaciones del clasificador y permitir una evaluación de cara a mejoras futuras.

1.3 Organización de la memoria

La memoria está organizada como sigue:

- Capítulo 1: Introducción: Su objetivo es presentar al lector la problemática presente y guiarle en los objetivos del trabajo.
- Capítulo 2: Estado del arte: Introducción a los usos presentes de las técnicas de clasificación usadas.
- Capítulo 3: Diseño y Desarrollo: Exposición detallada de la arquitectura del sistema completo, de la transformación de los datos para conseguir una distribución mejor modelable y la descripción del desarrollo de los diferentes back-end, además de las medidas de rendimiento usadas. Además, en este capítulo se detallan las características de las señales y los tipos de eventos a clasificar.
- Capítulo 4: Integración, pruebas y resultados: En este capítulo se realiza el análisis de los datos de entrada y el análisis de los resultados del Back-end. Comenzando con un análisis estadístico de los scores del Front-end, analizando los datos. Luego un análisis detallado de los resultados del Back-end, y por último un análisis de robustez en un escenario simulado de pocos datos de entrenamiento.
- Capítulo 5: Conclusiones y trabajo futuro: El fin de esta sección es sintetizar las conclusiones obtenidas en el desarrollo y vistos los resultados, además de formular las posibilidades de trabajo en el futuro.

2 Estado del arte

2.1 Clasificación de eventos mediante arquitectura Front-end - Back-end

En el estado actual de los sistemas de reconocimiento de idioma se suele añadir un back-end al final del sistema, con un doble propósito: cuando el conjunto de lenguas con el que se ha entrenado el modelo no coincide con el conjunto de lenguas objetivo, el back-end mapea las puntuaciones disponibles al nuevo espacio. Además, el back-end sirve como calibración para adaptar el espacio de las puntuaciones. [1]

Por lo tanto, en reconocimiento de idiomas la estructura Front-end – Back-end se usa frecuentemente, siendo el Front-end el que calcule ciertas puntuaciones en un primer paso de la clasificación usando características fonotácticas, y con las puntuaciones obtenidas, el Back-end clasifique finalmente las observaciones.

Además, en reconocimiento de idioma, un elemento esencial que proporciona el back-end es el poder usar varios subsistemas como elementos de entrada, que podrían ser generadores de puntuaciones, en diferente escala y de diferente tipo, donde la calibración final en esas puntuaciones del back-end es esencial. Y por otro lado el back-end realiza una fusión entre los diferentes reconocedores. [2]

2.1.1 Uso de back-end en reconocimiento de idiomas

En reconocimiento de idioma se usan un gran número de tipos de back-ends como, por ejemplo: [1]

- Generative Gaussian Back-end, es decir, un Back-end en el cual la distribución de las puntuaciones de lenguas es modelada mediante una distribución normal multivariada N (μ, Σ) estimándose sus parámetros mediante Maximum Likelihood (de ahora en adelante, ML). Es de los back-ends más simples de calcular, al ser en el caso de la gaussiana multivariada, el vector de medias μ la media de las muestras:

$$\mu = \frac{1}{n}\sum_{k=1}^{n} x_k \quad (1)$$

  y la estimación de la matriz de covarianza mediante ML:

$$\Sigma = \frac{1}{n}\sum_{k=1}^{n} (x_k - \mu)(x_k - \mu)^t \quad (2)$$

- Discriminative Gaussian Back-end, tras una estimación ML de las medias usadas inicialmente, y la matriz de covarianza común, se reestima iterativamente las medias para poder maximizar el criterio de Máxima Información Mutua (MMI).
- Fully-bayesian Gaussian Back-end, de la misma forma que el Generative Gaussian Back-end, se pretende modelar los datos con una gaussiana multivariada, sin

embargo, en vez de usar una estimación ML para calcular los parámetros del modelo, se integra de acuerdo con la información a priori, sobre todos los otros posibles parámetros.

- Generative Gaussian Mixture Back-end: Se amplía el modelo Gaussiano a una mezcla de gaussianas, compartiendo una matriz de covarianzas entre todas clases objetivo y calculando las medias de la mezcla de gaussianas mediante el algoritmo Expectation-Maximization (EM).

Estos son algunos de los modelos back-end usados en reconocimiento de idioma, que se han tomado como posibles candidatos a usar en el sistema implementado en este Trabajo de Fin de Grado.

3 Diseño y Desarrollo

3.1 Arquitectura del sistema

En este apartado, se describirá la arquitectura del sistema completo, centrándonos en la parte implementada por este TFG, pero empezando con un pequeño resumen de la extracción de características y primer paso en la clasificación del *Front-end*, de cuyos scores parte este TFG.

3.1.1 Front-end

Existen dos front-end, desarrollados e implementados ambos dos en "*Extracción de Características para señales temporales procedentes de sensores industriales, Álvaro Iglesias Arias"* [3] uno basado en características espectrales y otro basado en correlación.

3.1.1.1 ***Front-end por características espectrales***

El *front-end* por características espectrales está dividido en dos etapas claves: la extracción de características a partir de la información espectral, y un posterior cálculo de distancias respecto a unos centroides entrenados, que son las puntuaciones de entrada al *back-end*.[3]

3.1.1.1.1 Extracción de características

La extracción de características es el primer paso del *Front-end* y consta de los siguientes pasos:

- Se deriva la señal para quitar el offset de la señal y se realiza un filtrado paso bajo para quitar ruido de alta frecuencia sin información relevante de la señal.
- Posteriormente se enventana mediante una ventana rectangular con una superposición alta.
- De cada una de las ventanas se obtiene la transformada rápida de Fourier (FFT). De los coeficientes de la FFT se eligen mediante un análisis de variabilidad inter e intra clase, con el fin de escoger las características que más separan las clases y mejor definen una clase.

3.1.1.1.2 Distancia Euclídea y de Mahalanobis

Tras la extracción de características, se entrenan unos centroides por cada clase, de los cuales se obtiene una distancia.

Esta distancia, ya sea calculada mediante la distancia euclídea o mediante la distancia de Mahalanobis, proporcionan los scores con los cuales el *Back-end* tiene que clasificar.

3.1.1.2 *Front-end por correlación*

Por otro lado, se ha desarrollado un front-end por correlación, donde, previo cierto preprocesamiento de la señal y después de entrenar unas plantillas con la forma de cada uno de los eventos, se realiza una correlación en la señal con dichas plantillas.

Esta correlación ofrece picos en aquellos puntos donde la correlación es máxima, es decir donde la plantilla coincide con el evento buscado.

3.1.2 Transformación de datos

3.1.2.1 *Justificación*

Habiendo escogido un back-end con modelado gaussiano, es importante que los datos a modelar tengan una distribución estadística lo más cercana posible a la de la gaussiana. En audio se usan técnicas como pueden ser CVN (Cepstral Variance Normalization) o CMN (Cepstral Mean Normalization) que modifican la distribución estadística de los parámetros como la media o la varianza. Otro ejemplo de transformación de los datos es realizar una transformación fija de las variables, como por ejemplo tomar el logaritmo de los scores para que estén en el mismo dominio que la distribución Gaussiana.

Otro ejemplo de transformación es el feature warping que pretende modificar no sólo los parámetros estadísticos sino también la distribución de las características a corto plazo, para tener finalmente una distribución Gaussiana de media 0 y varianza unidad.

3.1.2.2 *Feature Warping*

Como se ha comentado antes, el feature warping es una técnica que modifica la distribución estadística de las características a corto plazo, con el fin de que la distribución completa mantenga esa distribución objetivo.

En el caso de este Trabajo de Fin de Grado, se busca una distribución Gaussiana en las características, por tanto, la distribución objetivo del feature warping es por lo tanto una distribución normal. Se ha implementado el artículo "*Feature Warping for Robust Speaker Verification* ", de Jason Pelecanos, [4] donde se propone una implementación de un algoritmo de Feature Warping.

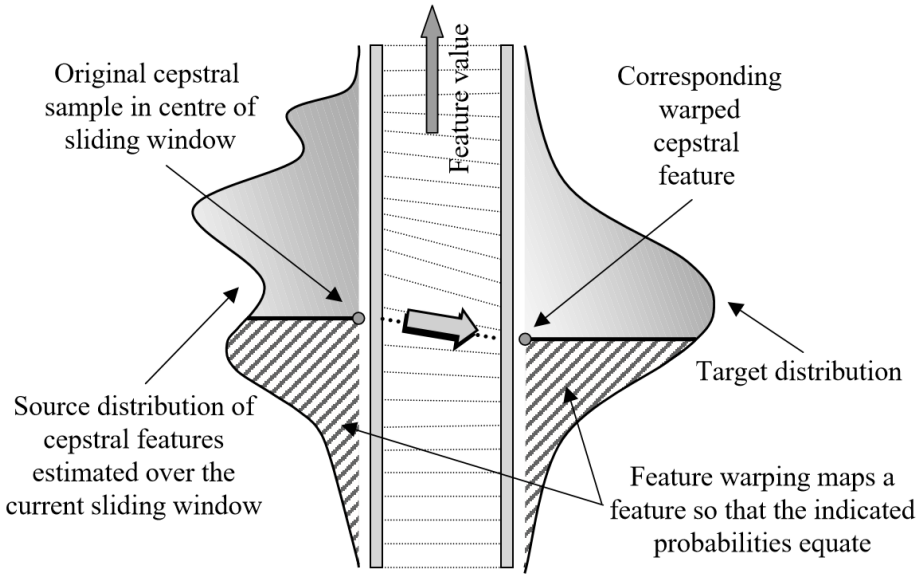

Figura 3-1: *En esta figura, extraída de "Feature Warping for Robust Speaker Verification", de Jason Pelecanos[4] se muestra el esquema general que sigue el feature warping, cambiando el valor de cada muestra dentro por el correspondiente en la distribución objetivo, con el fin de que las probabilidades de la distribución original y de la objetivo sean las mismas.*

La implementación del algoritmo del feature warping trata de, mediante una ventana deslizante ir mapeando la distribución original de la ventana a la distribución objetivo, según el esquema presentado en la **Figura 3-1** calculada previamente para ese tamaño de ventana y guardada en una *look-up table*:

$$\frac{N+\frac{1}{2}-R}{N} = \int_{z=-\infty}^{m} h(z)dz \quad (3)$$

Siendo N el tamaño de la ventana y R la posición del valor central de la ventana en la distribución original. De esta forma, se considera que la posición relativa del valor en su ventana es la estimación de su valor en la distribución original, al cambiarle su valor por el de la distribución gaussiana, se cambia la distribución de la señal a una normal de media cero y varianza unidad.

Es la forma de implementar la forma continua de la ecuación (4), en la cual $f(y)$ es la distribución medida de la ventana y $h(z)$ la función objetivo.

$$\int_{y=-\infty}^{q} f(y)dy = \int_{z=-\infty}^{m} h(z)dz \quad (4)$$

Con este algoritmo puede implementarse cualquier distribución objetivo, cambiando la función $h(\cdot)$ por la distribución que interese. En nuestras señales se puede comprobar en las **figuras 3-2 y 3-3** cómo la distribución de una de las clases se vuelve gaussiano con media cero y varianza unidad tras aplicarle *feature warping*.

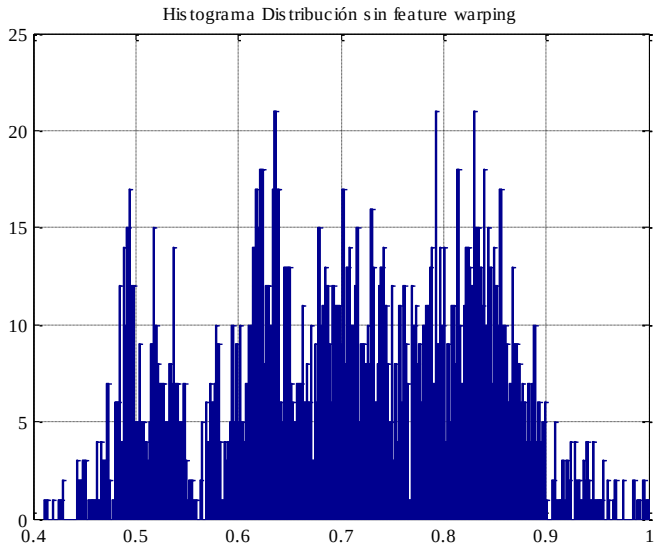

Figura 3-3

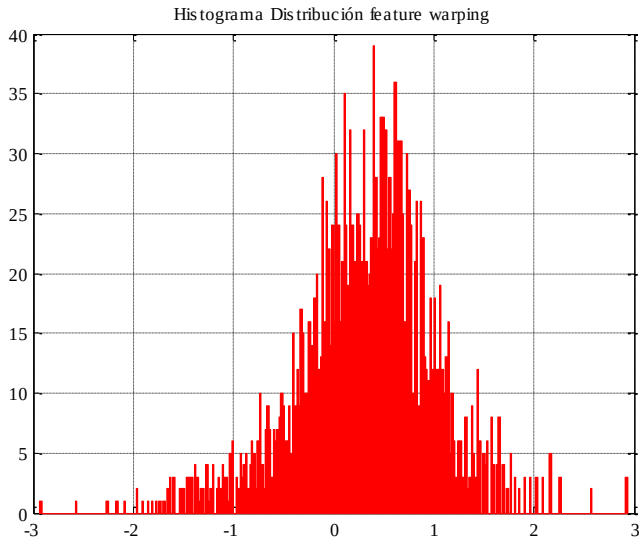

Figura 3-2

Figura 3-2 y 3-3 *Estas figuras representan a la izquierda, la distribución de las puntuaciones de los detectores a la entrada del back-end. A la derecha la distribución una vez aplicado el Feature Warping.*

Sin embargo, se ha comprobado que pese a que el feature warping funciona muy bien en señales de audio debido a sus características como se puede comprobar en estos coeficientes MFCC, donde se consigue que la señal tenga una distribución gaussiana además de conseguir media 0 y varianza unidad como se puede comprobar en la **figura 3-4**

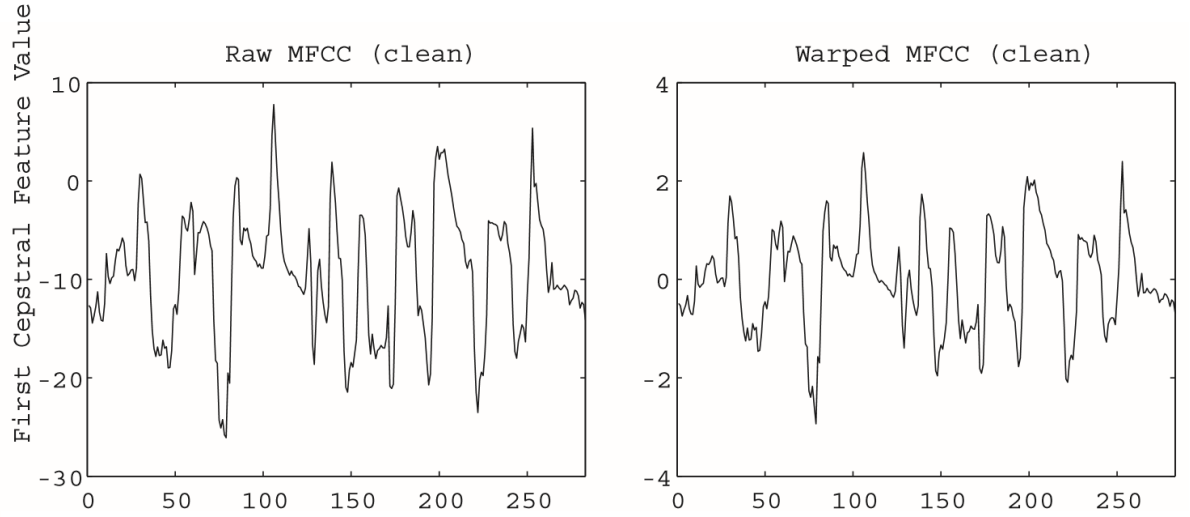

Figura 3-4

Figura 3-4: *Esta figura obtenida de "Feature Warping for Robust Speaker Verification", de Jason Pelecanos[4], muestra cómo funciona el Feature Warping otro tipo de señales, en este caso, el primer valor de los MFCC. Se puede ver cómo el algoritmo funciona muy bien con estas señales, consiguiendo media cero y varianza unidad sin perjudicar a la señal.*

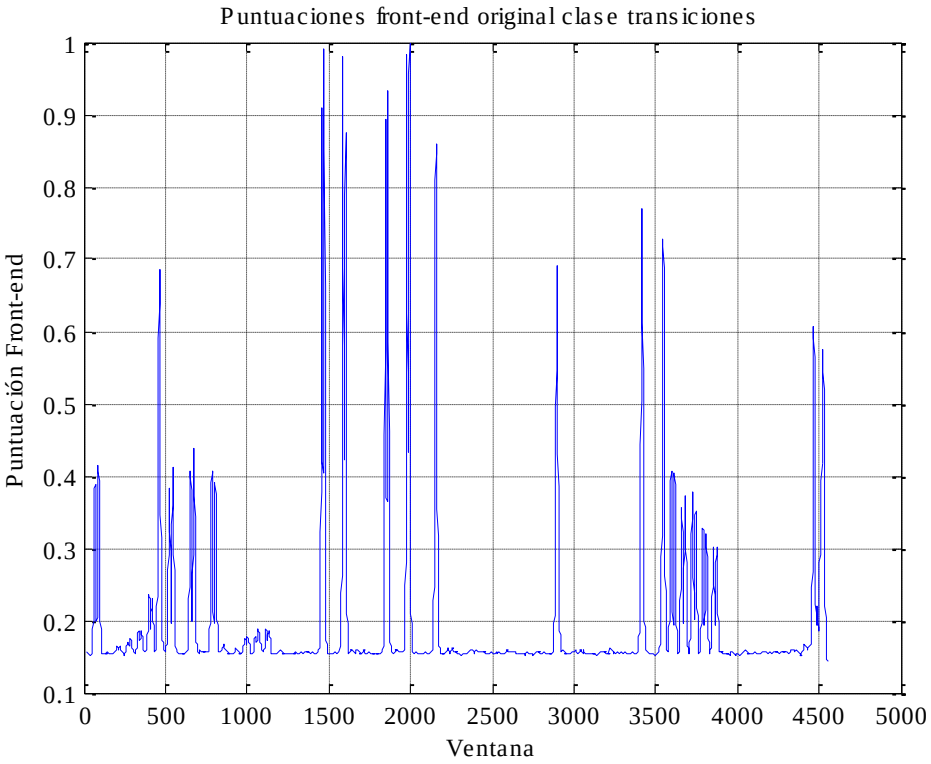

Figura 3-5

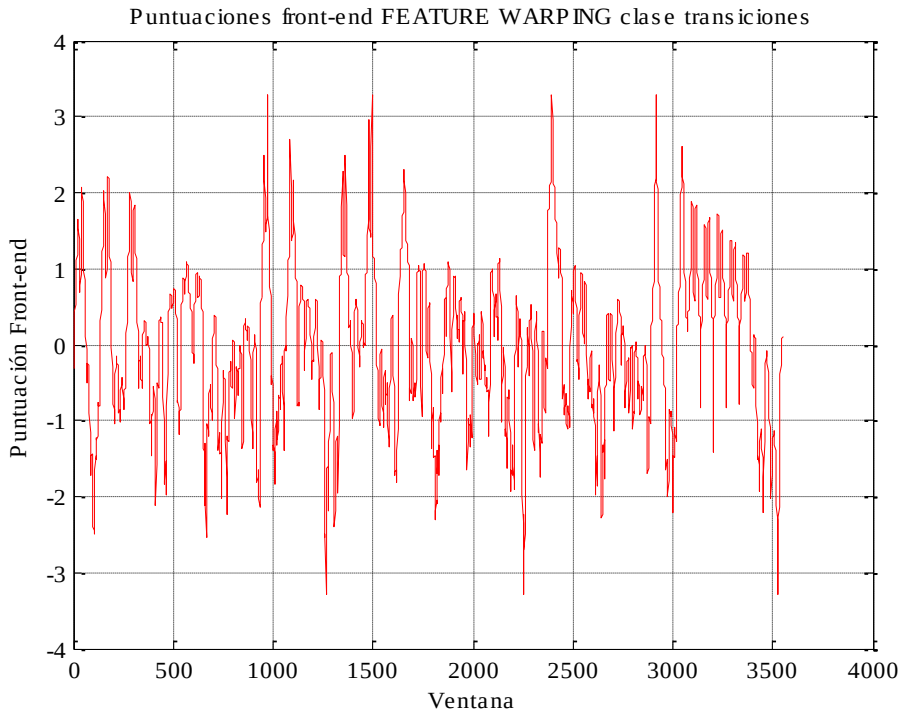

Figura 3-6: *En la figura de arriba Figura 3-5 se representan las puntuaciones del detector de la clase transiciones del Front-end. Los picos son definidos y permiten una buena clasificación. Sin embargo, en la figura de abajo Figura 3-6, se representa el resultado de usar Feature Warping con las puntuaciones anteriores, provocando que no se parte de la forma de los picos.*

Sin embargo, en el caso de las señales que llegan del front-end, que son puntuaciones de los clasificadores y no una señal dinámica en sí, el feature warping hace que las puntuaciones de los clasificadores se mezclen y que dichas puntuaciones discriminen peor entre eventos a detectar y ruido de fondo, como se verá en el capítulo 4.1 de esta memoria.

Se puede comprobar esto mismo viendo como las puntuaciones del front-end pierden nitidez, al cambiar su distribución con el feature warping:

En las figuras **3-5** y **3-6** se puede ver como la clase "transiciones" pierde la forma de la puntuación del clasificador del front-end, aunque este hecho se comprobará con mayor profundidad en el capítulo **4.1.**

3.1.2.3 *Transformación fija de variable*

Con el fin de acomodar la distribución de las puntuaciones del front-end al back-end gaussiano, además del feature warping, se han probado diferentes transformaciones fijas de variables que se han seleccionado de forma meramente heurística.

Al ser las puntuaciones del Front-end el resultado de una distancia, ya sea Euclídea o de Mahalanobis, esta distancia está comprendida entre 0 e infinito, el conjunto de números reales positivos, $x \in \mathbb{R}^+$

Sin embargo, la distribución gaussiana objetivo tiene su dominio $x \in \mathbb{R}$, para solucionar esto, podemos aplicar la transformación logaritmo, que cambia el dominio de las puntuaciones.

3.1.3 Back-end

3.1.3.1 *Justificación uso de back-end*

El uso de un back-end está justificado por varias razones en su uso en reconocimiento de idioma, principalmente para:

- Realizar un mapeo en caso de que los idiomas con los que se ha entrenado el modelo no coincidan con los idiomas de *test*. En nuestro caso, el detector por correlación del Front-end no tiene los mismos eventos que el Back-end, porque mientras que en el Back-end la clase desviaciones (picos en la señal) pueden ser picos hacia arriba o picos hacia abajo, en el detector de correlación se diferencian entre los picos hacia arriba y los picos hacia abajo, puesto que la forma de la plantilla es distinta y son por lo tanto dos clases distintas, con dos puntuaciones distintas. El Back-end es capaz de mapear estas dos puntuaciones a la de Desviaciones, y realizar una fusión.

- Calibrar las diferentes puntuaciones de los clasificadores del front-end. Las puntuaciones de cada uno de los detectores pueden estar en órdenes de magnitud diferentes y tener offsets, al usar un back-end no es necesario calibrar las

puntuaciones para que estén en el mismo rango dinámico, puesto que de esto se encarga el back-end.

- Fusionar las puntuaciones los diferentes subsistemas del front-end. La motivación de esto último sería la de usar las diferentes fortalezas de cada uno de los subsistemas, considerando que los errores de cada uno de ellos no está correlado.

En el caso particular de la clasificación de eventos de este trabajo, el back-end se ha usado para las tres razones comentadas más arriba.

Al tener dos subsistemas distintos en el front-end, el basado en características espectrales y el front-end por correlación, la fusión de ambos sistemas se realiza en el back-end mejorando los resultados de cada uno de ellos por separado.

Además, los eventos entrenados en el caso del front-end por correlación, como se verá más adelante en el punto 3.2, no se corresponden con los eventos del front-end de características espectrales, por tanto, el back-end realiza un mapeo de las puntuaciones.

Por último, al ser dos front-end completamente distintos, los scores de cada uno de ellos no tienen ninguna relación entre ellos, es necesaria una calibración, de la cual es encarga el back-end.

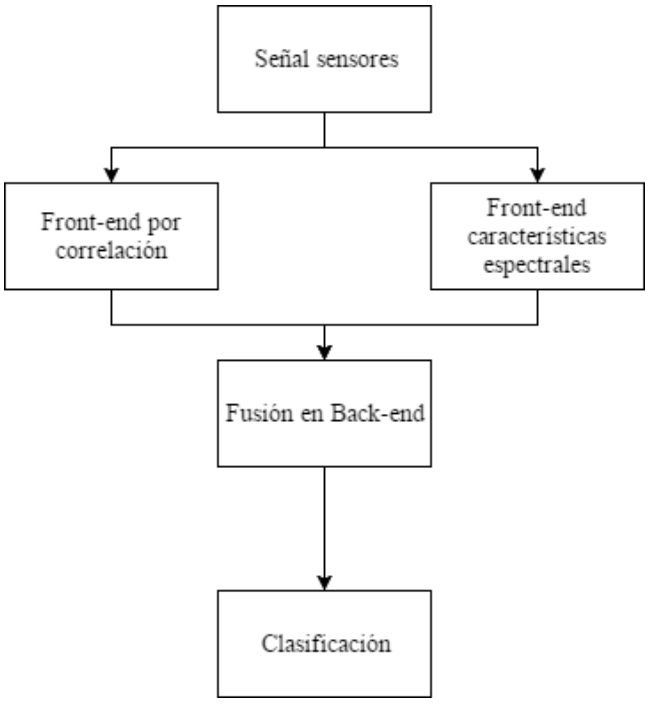

Figura 3-5: *En este esquema se representa el funcionamiento del Back-end en fusión. Se extrae de la señal de los sensores puntuaciones mediante los detectores de cada uno de los Front-end. Con esas puntuaciones se realiza una fusión en el Back-end que clasifica los eventos.*

3.1.3.2 ***Back-end baseline***

En una primera instancia se implementó un back-end baseline, cuyo objetivo es proporcionar una primera clasificación de eventos sobre el cual mejorar.

Es un back-end por máximos, clasificando cada nueva observación según sea la mayor puntuación del front-end para esa clase, lo que sería el inverso de la distancia en el front-end por características espectrales. Es la clasificación más simple, pero tiene la ventaja de que no requiere entrenamiento.

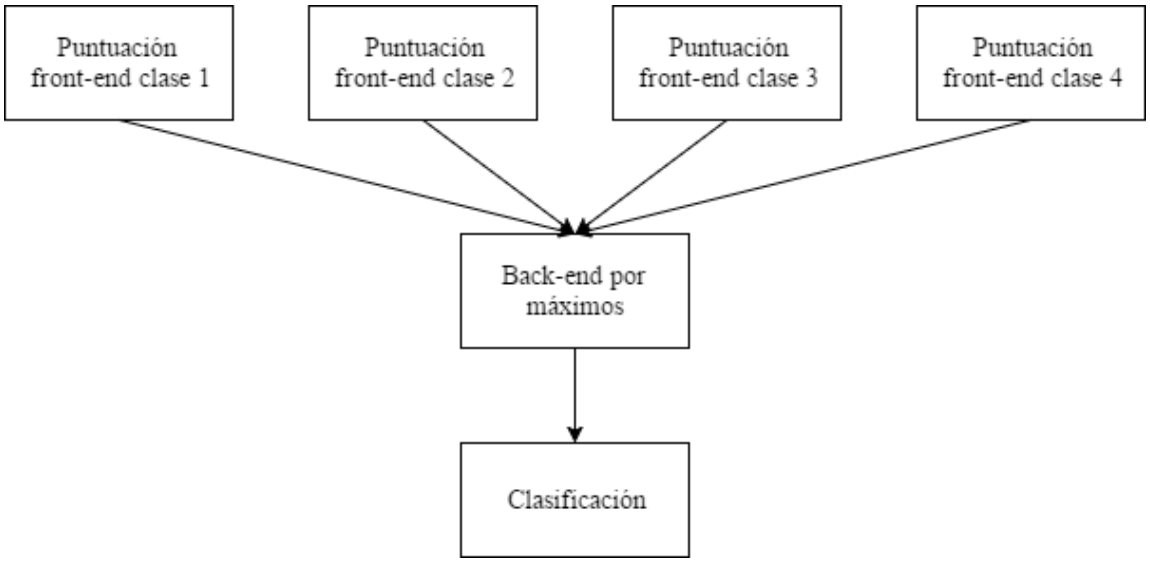

Figura 3-6: *En este diagrama se muestra el funcionamiento del back-end por máximos. Clasifica cada observación eligiendo la mayor puntuación de los detectores del Front-end.*

3.1.3.3 ***Back-end gaussiano estimado mediante ML***

Tras implementar un primer back-end baseline, se ha procedido a implementar otros back-end que mejoren el rendimiento del anterior.

Para modelar los datos se optó por usar un modelo gaussiano multivariado por las siguientes razones:

- En el ámbito de múltiples dimensiones, multivariado, existen limitadas distribuciones estadísticas con las que se pueden trabajar, siendo la gaussiana multivariada una de las más usadas en otros ámbitos como puede ser el reconocimiento de idioma.

- A la vista de las características estadísticas de las puntuaciones del front-end, se podía ver que un modelo gaussiano podría ser capaz de modelar los datos de entrenamiento, aunque estos no tuviesen una distribución gaussiana como tal.

- Existiendo la posibilidad de tener pocos datos de entrenamiento en este ámbito, usar un modelo más complejo podría causar problemas de sobre-entrenamiento.

La gaussiana normal multivariada $\mathcal{N}(\mu, \Sigma)$, está parametrizada por μ y Σ, que son los parámetros a estimar. Consideremos el conjunto μ y Σ como el conjunto de parámetros θ. Por lo tanto, estimar θ equivaldrá a estimar μ y Σ.

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \quad (5)$$

Supongamos además que nuestro conjunto de datos de entrenamiento está dividido entre sus *c* clases $\mathcal{D}_1, \ldots, \mathcal{D}_c$, y cada observación en $\mathcal{D}_j$ es obtenida independientemente.

El conjunto de datos $\mathcal{D}$ contiene n observaciones $x_1, \ldots, x_n$, por lo tanto, siendo θ los parámetros del modelo tenemos que:

$$p(\mathcal{D}|\theta) = \prod_{k=1}^{n} p(x_k|\theta) \quad (6)$$

Siendo $p(\mathcal{D}|\theta)$ la verosimilitud de θ respecto al conjunto de datos $\mathcal{D}$.

Por tanto, conseguir la *máxima verosimilitud* es obtener los parámetros $\hat{\theta}$ que maximicen $p(\mathcal{D}|\theta)$.

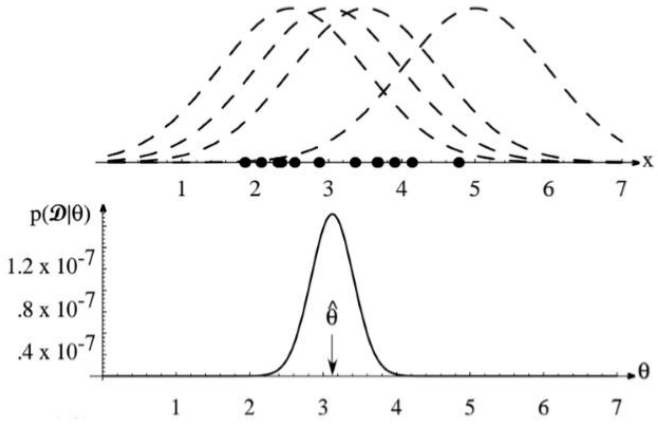

Figura 3-7: *En la figura superior (obtenida de Pattern classification, Duda, Richard O; Hart, Peter E; Stork, David G) se observan los datos de entrenamiento, supuestamente obtenidos de un de una distribución gaussiana cuyos parámetros* $\boldsymbol{\theta}$ *se desconocen. Se representan las posibles gaussianas a las que pueden corresponder estos parámetros. En la figura inferior se representa la gaussiana obtenida a partir de los parámetros* $\widehat{\boldsymbol{\theta}}$*, que maximizan la verosimilitud.[5]*

Esta maximización en la práctica se realiza con el logaritmo de la verosimilitud puesto que, al maximizar el logaritmo de la verosimilitud, se maximiza la verosimilitud en sí, al ser la función logaritmo una función monódica creciente, la ventaja de realizar esto es poder realizar las operaciones de forma analítica, y en la práctica no operar con valores excesivamente pequeños porque esto podría llevar a problemas numéricos de cálculo computacional. [5]

En el caso de la gaussiana multivariada, se obtiene analíticamente los parámetros $\hat{\theta}$ que maximizan la verosimilitud son:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \quad (1)$$

Siendo esta la media de los datos y

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n} (x_k - \mu)(x_k - \mu)^t \quad (2)$$

Siendo esta la estimación de la matriz de covarianzas.

Estos resultados son muy satisfactorios, pudiendo así, estimar directamente los parámetros θ de nuestro modelo obteniendo directamente la media y la covarianza de los datos de entrenamiento, por cada clase.

3.1.3.4 ***Back-end gaussiano estimado mediante modelo bayesiano***

En la estimación mediante máxima verosimilitud, se asumen que los parámetros del modelo a estimar son *desconocidos*, pero *fijos*. Por tanto, su mejor estimación es la que maximiza la probabilidad de obtener las muestras ya observadas.

Sin embargo, los métodos Bayesianos consideran que los parámetros del modelo a estimar son *variables aleatorias* con una distribución conocida *a priori*, además de *desconocidos*. Las nuevas observaciones convierten esto en una densidad *a posteriori*, cambiando los valores de los parámetros. La densidad *a posteriori* según van aumentando las muestras observadas tiende a alcanzar un pico alrededor de los valores verdaderos de los parámetros.

Por lo tanto, la estimación Bayesiana será idéntica a la estimación ML cuando el número de muestras observadas sea grande, sin embargo, si el número de observaciones se reduce, habrá cierta incertidumbre en los parámetros θ a estimar.

En el caso gaussiano, al igual que en ML, se asume que se conoce la función $p(\boldsymbol{x}|\theta)$, es decir, una gaussiana multivariada de la cual se desconocen los parámetros θ. Además, se conoce la información de densidad *a priori* de los parámetros θ es decir, $p(\theta)$.

Tras varias observaciones, $p(\theta)$ se convierte en $p(\theta|\mathcal{D})$, la densidad *a posteriori*, que se espera que de una estimación precisa al cabo de muchas muestras del valor verdadero de θ.

Nuestro objetivo sin embargo es obtener $p(\boldsymbol{x}|\mathcal{D})$. Es decir, la probabilidad de que la observación $\boldsymbol{x}$ pertenezca al conjunto de datos $\mathcal{D}$, siendo $\mathcal{D}$, el conjunto de datos de una clase en particular.

Esto puede ser calculado a partir de la densidad a posteriori integrando las marginales de θ

$$p(\boldsymbol{x}|\mathcal{D}) = \int p(\boldsymbol{x}|\theta)p(\theta|\mathcal{D})\, d\theta \quad (7)$$

La operación principal es por tanto obtener la densidad a posteriori $p(\theta|\mathcal{D})$, que mediante la fórmula de Bayes tenemos:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)\, d\theta} \quad (8)$$

Donde $p(\mathcal{D}|\theta)$ es la función de verosimilitud y $p(\theta)$ es la información *a priori*.

La función de verosimilitud $p(\mathcal{D}|\theta)$ puede ser obtenida mediante:

$$p(\mathcal{D}|\theta) = \prod_{k=1}^{n} p(\boldsymbol{x}_k|\theta) \quad (9)$$

por la asunción de independencia.

De estas ecuaciones se puede comprobar la relación de esta estimación con respecto a la ML, en el supuesto de que $p(\mathcal{D}|\theta)$ tuviese un pico en $\theta = \hat{\theta}$, entonces $p(\theta|\mathcal{D})$ también tendría un pico en ese punto. Por lo tanto $p(\boldsymbol{x}|\mathcal{D})$ sería equivalente a $p(\boldsymbol{x}|\hat{\theta})$, es decir la solución de máxima verosimilitud.
La información de densidad *a priori* de θ en caso de la gaussiana multivariada suele modelarse mediante una distribución Gaussian-Wishart, siendo la Gaussiana de la distribución la que modela μ y la distribución Wishart la que modela Σ. La distribución Wishart es una versión en multiples dimensiones de la función Gamma, siendo esta la usada en el caso univariado, para modelar la distribución *a priori* de los parámetros a estimar. [6]

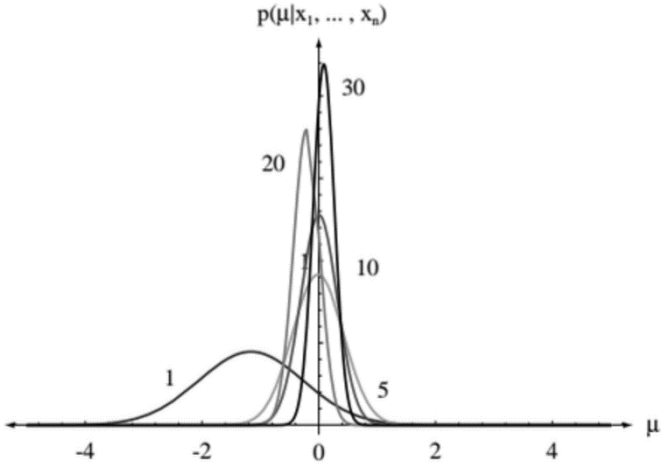

Figura 3-8 [4]: *Según aumenta el número muestras de entrenamiento, la incertidumbre disminuye, estimándose de forma más precisa μ*

El uso de estas distribuciones en la información *a priori* de los parámetros es la que proporciona cierta incertidumbre en el valor.

En el caso de μ, por ejemplo, se puede comprobar cómo según aumentan las observaciones, la distribución gaussiana toma una forma de mayor pico, alrededor de un valor, donde se situaría $\hat{\mu}$, el valor real de μ. Por lo tanto, según se aumente el número de muestras

observadas, la incertidumbre disminuirá, y la estimación Bayesiana y la de Máxima Verosimilitud coincidirán.

Sin embargo, en un escenario con pocos datos de entrenamiento, esta incertidumbre en la estimación de los parámetros proporciona una mayor robustez en el entrenamiento, al permitir una cierta incertidumbre en la estimación de los parámetros. Por tanto, está justificado el uso de la estimación Bayesiana para un escenario de escasez de datos de entrenamiento. [5]

Sin embargo, queda la problemática de integrar las marginales como en (6), $p(\boldsymbol{x}|\mathcal{D}) = \int p(\boldsymbol{x}|\theta)p(\theta|\mathcal{D})\, d\theta$ para hallar $p(\boldsymbol{x}|\mathcal{D})$, la probabilidad de que un nuevo valor pertenezca a una clase en cuestión.

Por suerte, en el caso de una gaussiana multivariada, con la información *a priori*, estimada mediante una Gaussian-Wishart, la integración en (6) tiene un resultado analítico:

$$p(x|\mathcal{D}) = \frac{\Gamma\left(\frac{N+1}{2}\right)}{\Gamma\left(\frac{N+1-d}{2}\right)} \left|\frac{NS^{-1}}{\pi(N+1)}\right|^{\frac{1}{2}} \left(\frac{N(x-\bar{x})^T S^{-1}(x-\bar{x})}{N+1} + 1\right)^{-(N+1)/2} \quad (8)$$

Siendo $\Gamma(\cdot)$ la función Gamma, S la matriz de dispersión, N el número de observaciones de entrenamiento y d el número de dimensiones. [7]

Esta distribución, $p(x|\mathcal{D})$, también llamada la distribución predictiva a posteriori, coincide en este caso, con una T-Student multivariada, con los parámetros hallados anteriormente.

3.1.3.4.1 T-Student

Siendo el resultado final del cálculo anterior una T-Student, merece la pena detenerse para observar las características de esta distribución.
La T-Student, varía con el parámetro N, también llamado grados de libertad. Cuanto mayor es este parámetro entero, más parecido tiene con la gaussiana normal. En el caso de una dimensión, a partir de N>75 para el caso de una única dimensión las curvas son indistinguibles a simple vista.

Por otro lado, la T-Student se diferencia de la distribución Normal en que presenta unas colas más altas. Esto se corresponde con la incertidumbre que existe en los parámetros a la hora de estimar el modelo Gaussiano.

De hecho, se podría decir que la T-Student es el caso general, mientras que la Gaussiana es un caso particular de la T-Student donde N tiende a infinito, donde desaparece toda incertidumbre.

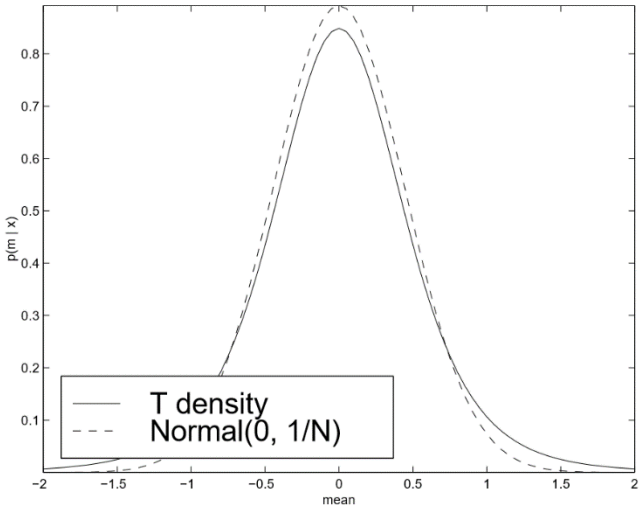

Figura 3-10 [7]: *En esta figura obtenida de "Minka, Inferring a Gaussian Distribution" [6] se ve como la distribución normal presenta unas colas más bajas respecto a la T-Student, cuando N es bajo. En este caso N=5.*

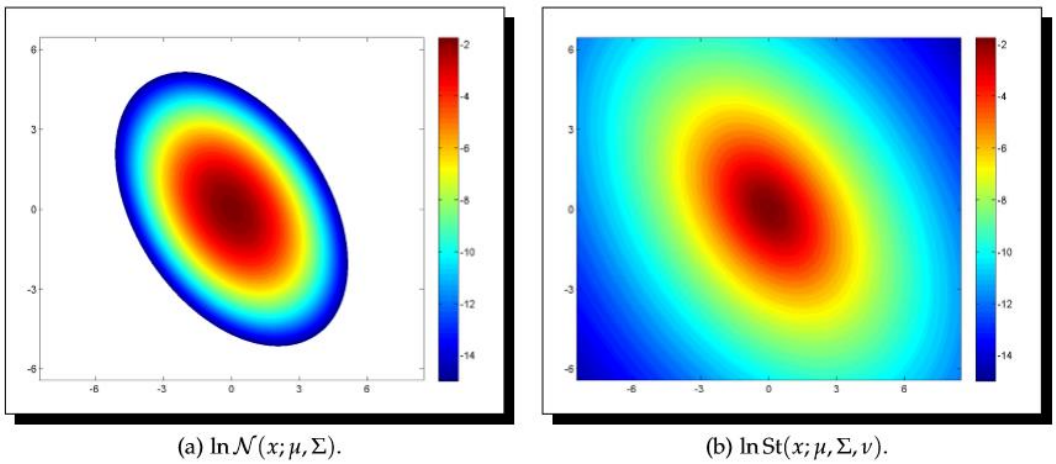

Figura 3-9 [8]: *En estos mapas de calor en escala logarítmica, se puede comprobar como la T-Student en dos dimensiones presenta colas más altas respecto a la Distribución Normal Multivariada. Las zonas en blanco están por debajo de -15 dB. Por lo tanto, la distribución T amplía las posibilidades de modelar datos más alejados de la media. En el caso de tener pocas muestras, esta incertidumbre permite que una media mal calculada mediante ML por falta de datos no estropee el modelado.*

Estas colas más altas hacen que la T-Student sea capaz de modelar mejor distribuciones estadísticas del mundo real, donde se presentan a menudo outliers, además la hace más robusta frente a un escenario con pocos datos de entrenamiento como puede ser la problemática a la que se enfrenta este TFG. Esto está representado en las **Figuras 3-11 y 3-12 [7]**

3.1.4 Medidas de rendimiento

Para evaluar el rendimiento total del sistema a la salida del back-end se han implementado distintas medidas de rendimiento, tanto numéricas como gráficas.

3.1.4.1 ***Accuracy, Precision, Recall y Fscore***

En reconocimiento de patrones a la hora de evaluar un sistema de clasificación binaria, se dividen los resultados tras la clasificación en:

- True positives (tp): De los resultados obtenidos como positivos, los que han sido clasificados correctamente como positivos.
- False positives (fp): De los resultados obtenidos como positivos, los que han sido clasificados incorrectamente como positivos (al ser en verdad negativos).
- True negatives (tn): De los resultados obtenidos como negativos, los que han sido clasificados correctamente como positivos.
- False negatives (fn): De los resultados obtenidos como negativos, los que han sido clasificados incorrectamente como negativos (al ser positivos).

Con estos datos se pueden implementar medidas de rendimiento más o menos eficaces como:

1. *Accuracy*: es el número de predicciones correctas dividido por el número total de predicciones hechas:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad (9)$$

   El problema de esta medida de rendimiento es que cuando las clases están desbalanceadas, ofrece resultados poco precisos y confusos.

2. *Precision*: es el número de predicciones positivas correctas dividido por total de predicciones positivas:

$$Precision = \frac{tp}{tp + fp} \quad (10)$$

   Indica en cierto modo la exactitud del clasificador, un valor bajo de precisión indica un gran número de falsos positivos.

3. *Recall*: El número de predicciones positivas dividido por la cantidad total de valores positivos en la clase de *test*.

$$Recall = \frac{tp}{tp + fn} \quad (11)$$

   También es llamado sensibilidad. Un valor bajo de *recall* indica muchos falsos negativos. Indica la completitud del clasificador.

4. *Fscore*: es una balanza entre el *Precision* y el *Recall*, la media armónica:

$$Fscore = 2\,\frac{Precision * Recall}{Precision + Recall} \quad (12)$$

El *Fscore* puede ponderar de la misma forma *Precision* y *Recall* como en la fórmula anterior o darle más importancia a uno que al otro, siendo β el factor de ponderación:

$$Fscore_{\beta} = (1 + \beta^2)\frac{Precision * Recall}{\beta^2 Precision + Recall} \qquad (13)$$

Se ha usado la fórmula (12) es decir, β = 1 para darle el mismo peso al *Precision* que al *Recall*.

El *Fscore* será la medida de rendimiento que más se usará en la evaluación del sistema, debido a que las clases están muy desbalanceadas como se verá más adelante.

3.1.4.2 ***Matriz de confusion***

Otra forma de evaluar el rendimiento de forma visual es mediante una matriz de confusión, en ella se representan los valores clasificados respecto al *ground truth*, según la estructura de la **Tabla 3-1**.

En esta matriz de confusión, los valores de la diagonal principal de la matriz serían los valores que han acertado al clasificar, y los valores fuera de esa matriz, en los que el clasificador se ha equivocado.

De esta forma es fácil ver en qué se ha confundido el clasificador, y entre qué clases se ha confundido.

Esta matriz de confusión es generalizable a un clasificador con varias clases, con una matriz de *c* x *c* donde *c* es el número de clases con el que trabaja el clasificador. Sigue manteniendo el mismo comportamiento, estando en la diagonal principal de la matriz el número de observaciones que se han clasificado por cada clase correctamente.

3.2 Base de datos

3.2.1 Características de las señales y tipos de eventos a clasificar

Este sistema está diseñado para clasificar los diferentes tipos de eventos que puede tener la señal captada por ciertos sensores que son sensibles a la variación de cierto material X en la pieza a analizar al final de la cadena de producción.

		Clasificación	
		Positivo	**Negativo**
Ground Truth	**Positivo**	True positives (tp)	False negatives (fn)
	Negativo	False positives (fn)	True negatives (tn)

Tabla 3-1: Matriz de confusión binaria

Esta sería una señal típica usada para el entrenamiento del clasificador, y los diferentes eventos a clasificar etiquetados diferentemente.

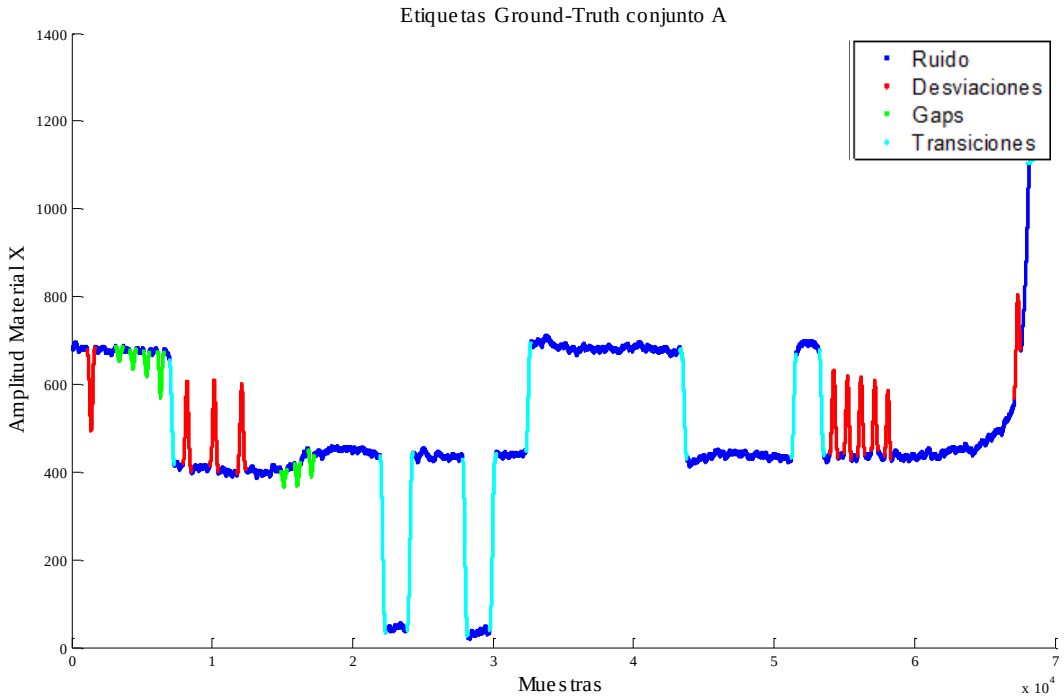

Figura 3-11: *En esta figura se muestra una señal de amplitud de material X, parte del conjunto A, según la recoge el sensor, con los diferentes eventos etiquetados.*

Se pueden diferenciar los siguientes eventos en la **Figura 3-13**:

- Ruido: Son las zonas de la señal en las cuales no existe ningún evento aparte del ruido, el cual tiene bastante amplitud debido a la naturaleza del material X y del sensor sensible a este material.

- Desviaciones: Desviaciones en la concentración del material X que generan picos en la señal del sensor.

- Huecos o *Gaps*: Pequeños huecos en la pieza que provocan una pequeña desviación hacia abajo en la señal.

- Transiciones: Transiciones entre zonas con diferente concentración del material X a medir.

Cada uno de los eventos tiene componentes espectrales y forma temporal distinta, de cuyo modelado se encarga el Front-end, ámbito del cual no se encarga este TFG.

3.2.2 Salida del Front-end y entrada al Back-end

Después de la obtención de puntuaciones realizada por el Front-end, a cuya problemática descrita brevemente en el apartado 3.1.1 no se enfrenta este TFG, que parte directamente de los scores allí calculados.

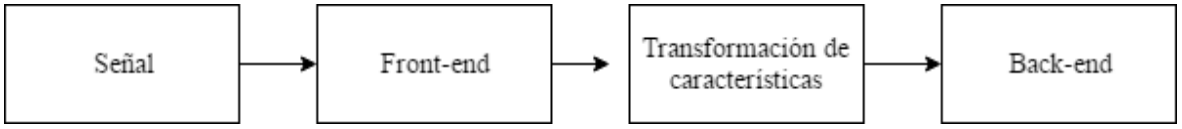

Figura 3-12: *Esquema del sistema completo*

En el caso del front-end basado en características espectrales, por cada nueva observación, el front-end da cuatro puntuaciones, una por cada tipo de evento. Estas puntuaciones son altas en caso de que el front-end detecte el evento para el que está entrenado en esa observación y bajas en caso de lo contrario.

Una figura con estas puntuaciones puede encontrarse en el Anexo A: **Figura Anexo-1.**

En el caso del front-end por correlación su comportamiento es similar,
aunque con diferentes eventos, que la propiedad del back-end de mapear las clases de entrada con las de la salida hace que no haga falta hacer ningún tipo de calibración o fusión además de la que se hace en el propio back-end.

Una figura con las puntuaciones del front-end por correlación se puede encontrar en el Anexo A: **Figura Anexo-2.**

3.2.3 Conjunto de datos

En la base de datos de la que se dispone hay 30 inspecciones distintas de la misma pieza. Estas 30 inspecciones son las únicas señales que han estado disponibles durante la mayor parte del tiempo de este trabajo. Este conjunto de señales es denominado: grupo A.

Un ejemplo de estas señales, todas idénticas salvo el ruido, está representado en la **figura 3-14** de la página anterior

Además, se incorporaron en última instancia señales de la inspección de otra pieza distinta al final del trabajo, con una distribución y forma de los eventos distintas. Este conjunto es el grupo B, representada en la **Figura 3-15**.

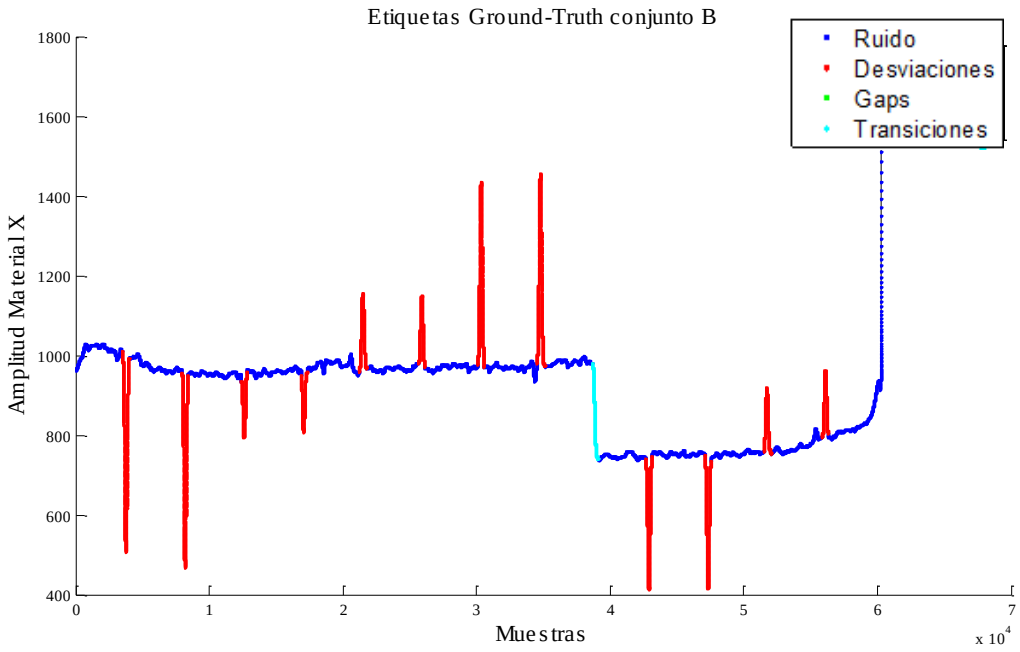

Figura 3-13: *Esta señal del conjunto B, tiene Desviaciones de diferentes tamaños, solo una transición y no tiene gaps, además de un ruido con más potencia. Al ser tan diferente del conjunto A, la tarea de la clasificación es más complicada.*

4 Integración, pruebas y resultados

4.1 Análisis estadístico de los detectores del Front-end

4.1.1 Visualización con reducción de dimensión

Con el fin de observar cómo se modelan los datos mediante una distribución gaussiana, se ha procedido a crear una visualización de ellos en dos dimensiones, ya que no es posible la visualización en las cuatro dimensiones en caso del esquema básico, o las nueve en caso de la estrategia de fusión.

En esta forma de visualización en dos dimensiones, se representa para cada observación de *test* (en cross-validation), su valor en dos de las dimensiones del clasificador, además de la gaussiana que ha sido entrenada previamente con los datos de entrenamiento.

4.1.1.1 ***Sin transformación de datos***

Como era de esperar, cada tipo de evento tiene una puntuación en su evento en concreto, por ejemplo, en este caso, las observaciones de las transiciones puntúan alto en la dimensión del Front-end de las transiciones y lo mismo ocurre en el caso de los *gaps* como se puede comprobar en la **Figura 4-1**

Los eventos que peor están separados son los *gaps* y el ruido, debido a que los *gaps* a menudo son de tan pequeña amplitud que se pueden confundir con zonas de ruido.

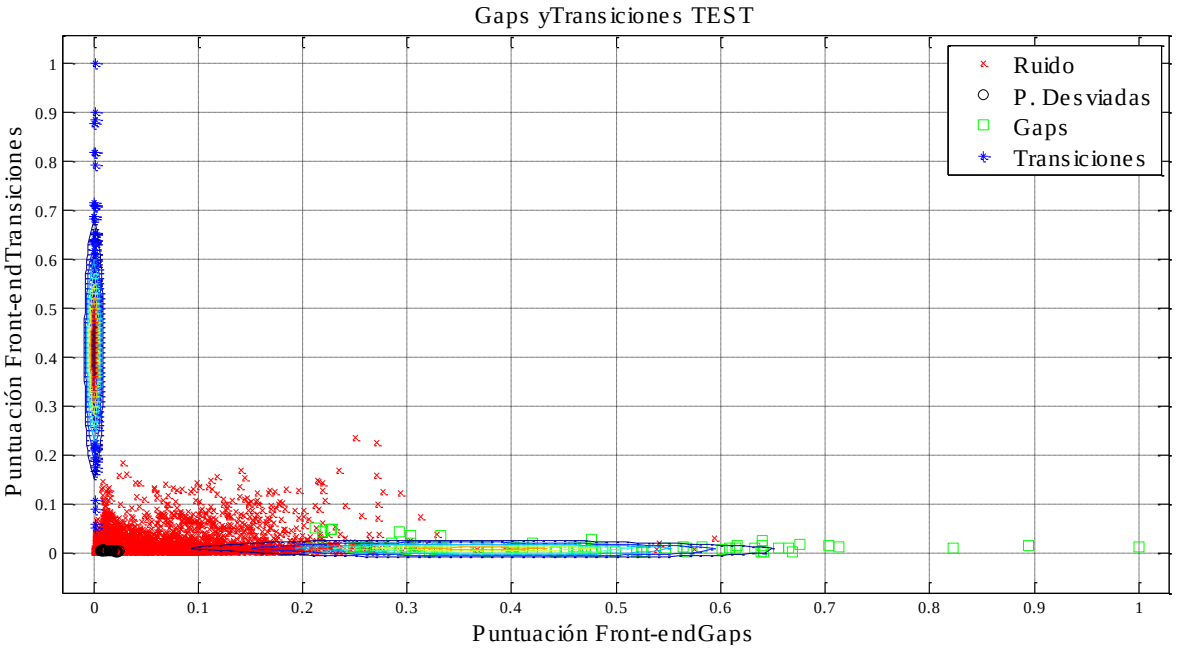

Figura 4-1: *En esta figura se representa cada observación etiquetada, con las puntuaciones de dos de los detectores del Front-end, en este caso el detector de huecos (o gaps) contra el de las Transiciones en el conjunto de Test. Además se representa un back-end bidimiensional entrenado previamente con los datos de entrenamiento.*

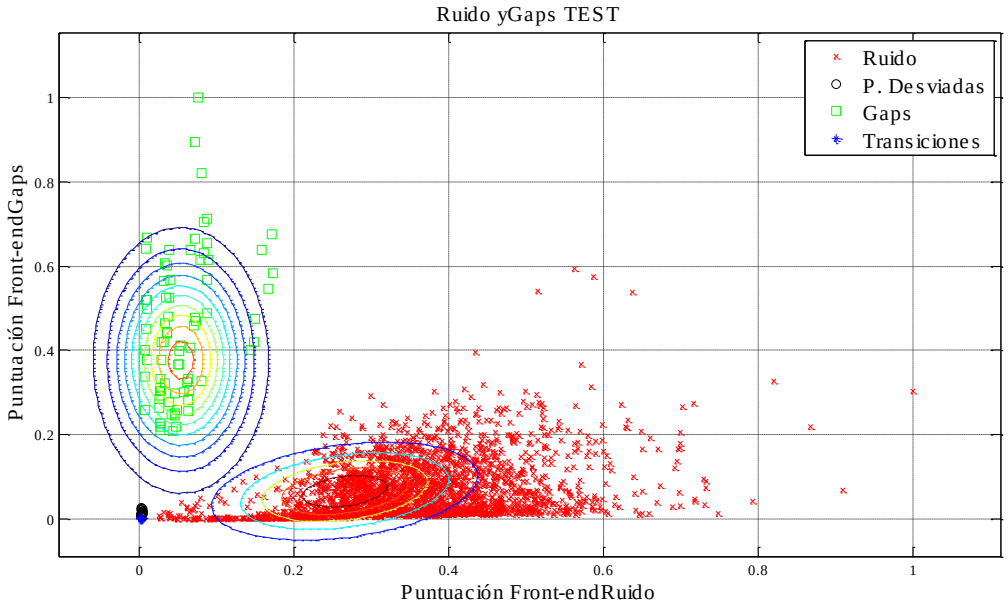

Figura 4-2: *Debido a la naturaleza de los gaps, huecos muy pequeños, también puntúan alto en el detector de ruido, y el ruido puntúa alto en el detector de gaps, por lo que su forma no es tan lineal. Aun así, los datos están separados en estas dos dimensiones.*

En la **Figura 4-2** se puede comprobar como los *gaps* y el ruido tienen un aspecto mucho menos lineal, debido a que ambos tienen cierta puntuación en la dimensión del contrario.

En cambio, los otros eventos, las desviaciones y las transiciones, no tienen ningún valor en esas dimensiones, estando muy bien separados los datos.

Sin embargo, la forma lineal de la distribución no se adapta bien a la gaussiana, de ahí la motivación de realizar transformaciones a los datos.

4.1.1.2 *Transformación fija de variable*

Como se puede comprobar en las figuras anteriores, las puntuaciones del Front-end son distancias, esta distancia está comprendida entre 0 e infinito, el conjunto de números reales positivos, $x \in \mathbb{R}^+$.

Sin embargo, la distribución gaussiana objetivo tiene su dominio $x \in \mathbb{R}$, siendo necesario un cambio de variable para ajustarse mejor a la distribución.

Como en el ejemplo de la **Figura 4-3**, se muestra en la puntuación del front-end de las transiciones frente al de los *gaps*, pero tomando logaritmos. Como era de esperar, cada tipo de evento puntúa mejor en su dimensión.

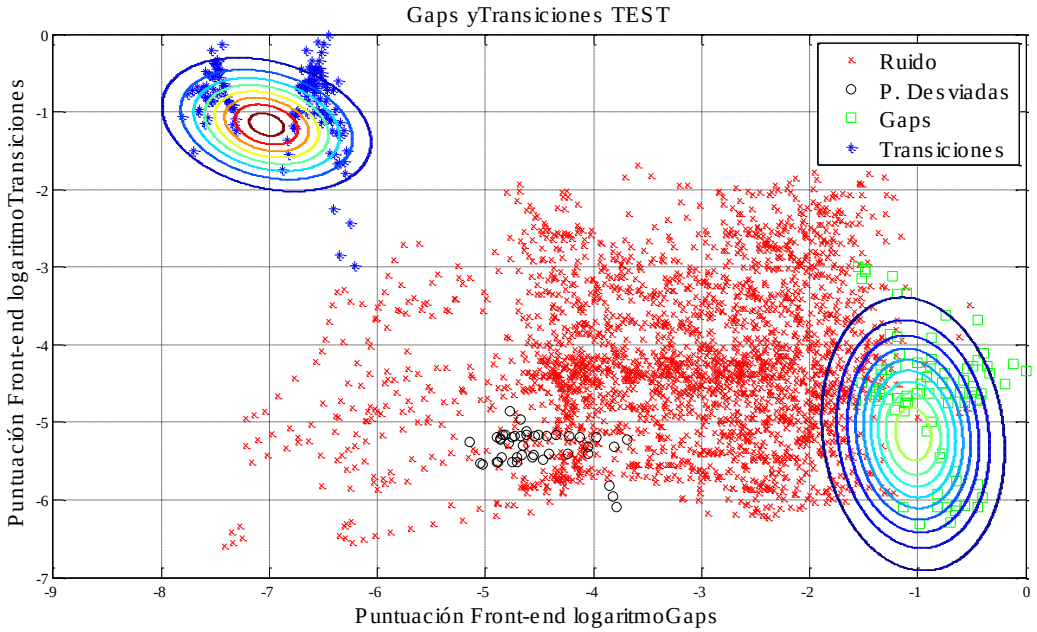

Figura 4-4: *Es el ejemplo idéntico a la **Figura 4-1**, pero usando la transformación logaritmo. En estas dimensiones, los datos de eventos transiciones y gaps están muy bien separados, pero además se consigue gracias a la transformación un mejor modelado de los datos.*

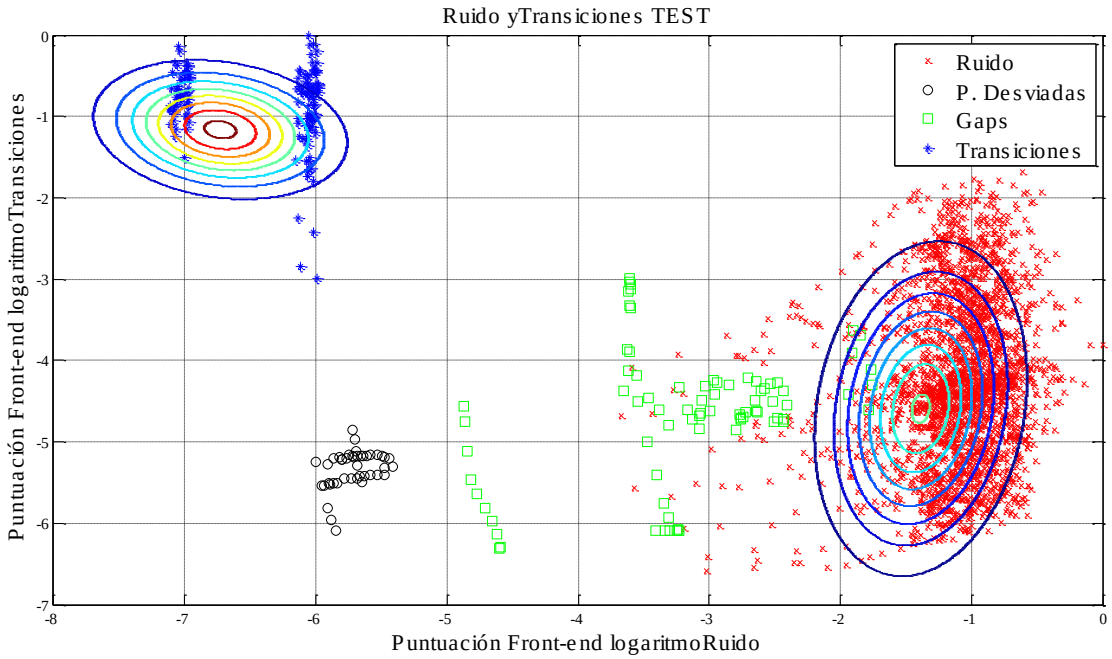

Figura 4-3: *En este ejemplo, se representan las puntuaciones del detector de Transiciones frente al detector de Ruido. Se puede ver cómo están los eventos están muy separados y el modelo gaussiano se ajusta relativamente bien a los datos.*

Cabe destacar que, aunque los eventos *gaps*, ruido y desviaciones parecen mezclados en la **figura 4-3**, en realidad son solo dos de las dimensiones, por ejemplo, en otras dimensión, las

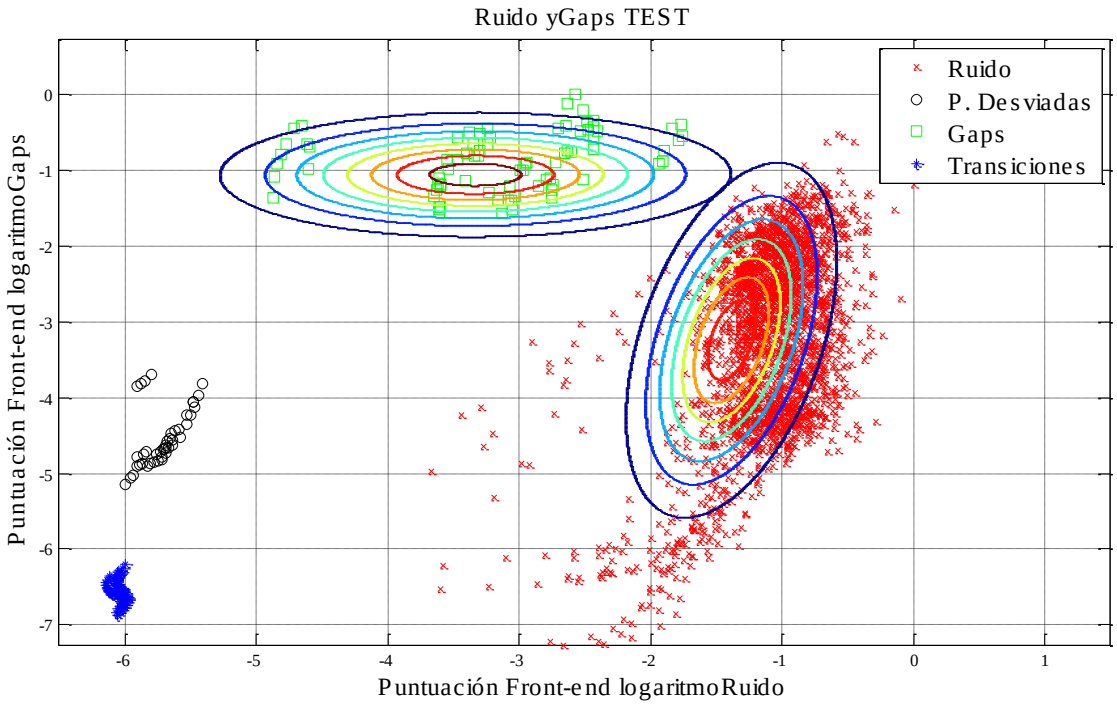

Figura 4-5: *Esta figura muestra la puntuación del detector ruido frente al detector gap. Los datos están separados, y las gaussianas se ajustan bien. Se puede ver, sin embargo, como los gaps tienen ciertas agrupaciones de datos, o clusters.*

desviaciones están completamente separadas de los otros eventos, como se puede apreciar en la **figura 4-4**

Es por ello que es importante realizar la clasificación con un back-end multidimensional, para aprovechar todas las dimensiones en las que los eventos se diferencian entre sí.

Por otro lado, los *gaps* y el ruido, como hemos dicho antes, son los eventos que más mezclados están, debido a la gran similitud entre ellos. Los *gaps* además están divididos en diferentes *clusters*, estas agrupaciones son las diferentes observaciones pero que coinciden con el mismo gap en la señal, como se observa en la **figura 4-5.**

El tener un conjunto muy limitado de entrenamiento en este trabajo, provoca este efecto, no siendo así si se tuviese una mayor variedad de eventos, de diferentes tamaños, que cubrirían los huecos entre los *clusters.* Lo mismo es verdad para las transiciones y desviaciones de material X.

La transformación del logaritmo consigue aun así que las observaciones se agrupen, y las gaussianas parecen modelar mejor los datos, como luego se comprobará con los resultados en F-score.

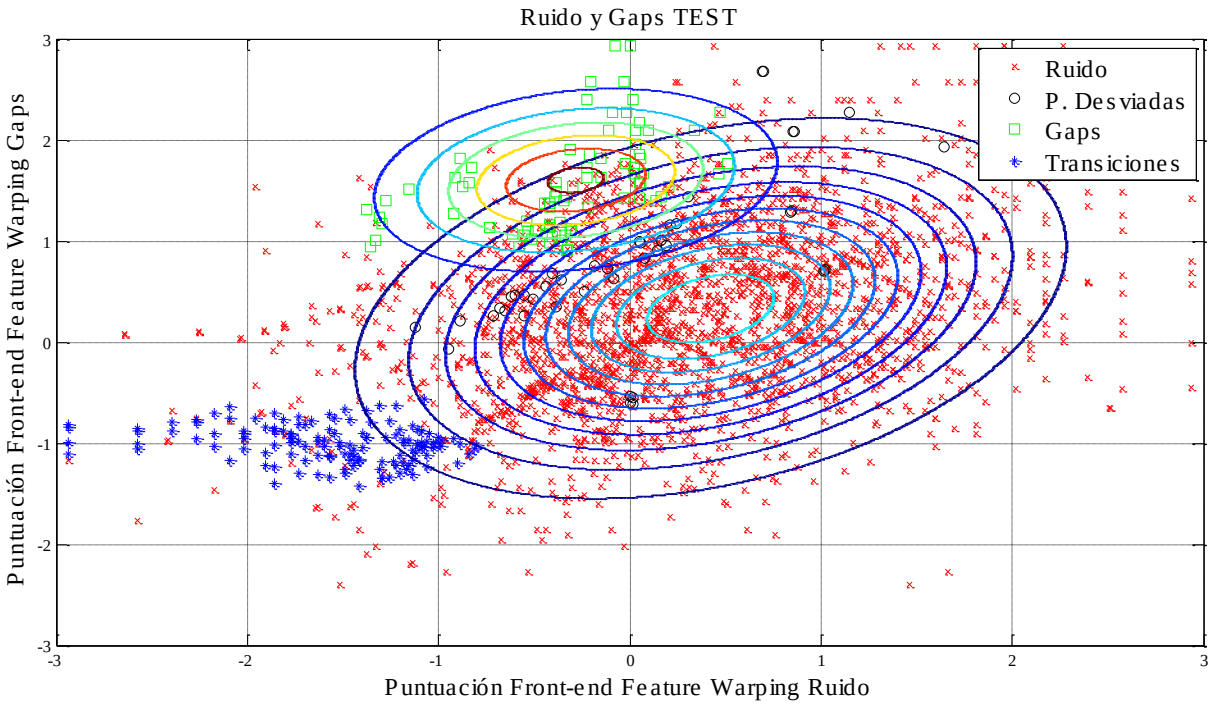

Figura 4-6: *Usando feature warping, el ruido y los gaps están completamente mezclados. Aunque la distribución de los datos tiene una forma más gaussiana y es más fácil modelarla, empeora la discriminación del clasificador.*

4.1.1.3 ***Feature Warping***

Por último, se ha realizado esta prueba usando feature warping. Como se vio en el apartado **3.1.2.2**, el Feature Warping pese a realizar una buena gaussianización, mezcla los eventos en vez de separarlos y se pierde capacidad de discriminación.

En este caso por ejemplo en el que se observa el comportamiento del ruido frente a los *gaps*, se puede ver cómo, aunque las observaciones de distribuyen de una forma más gaussiana, el poder de discriminación del clasificador disminuye enormemente, siendo imposible separar los eventos correctamente. Esta es la razón por la que finalmente el Feature Warping no ha sido útil para este trabajo, puesto que es más interesante tener una mejor separación entre los eventos, que ajustarse mejor a los datos con el modelo, como es visible en la **figura 4-6**.

Además, el uso de transformaciones de variable fija ha cubierto la necesidad de poder ajustar el modelo a los datos correctamente, dando muy buenos resultados de clasificación.

4.2 Resultados Back-end

Una vez diseñado e integrado todo el sistema y definidos las medidas de rendimiento, se procede a evaluar los resultados del Back-end.

Durante la mayor parte de la duración del trabajo, se tuvo la única base de datos de señales, el grupo A; por lo tanto, las pruebas se hicieron en validación cruzada. Las señales eran inspecciones distintas de la misma pieza, por tanto, la forma de la señal y los eventos son exactamente los mismos, cambiando el ruido de carácter aleatorio.

El procedimiento en validación cruzada es mantener una señal fuera del grupo de entrenamiento, entrenar el clasificador con esas señales de entrenamiento y después evaluar el rendimiento con la señal que se ha mantenido fuera del conjunto. Este procedimiento es llamado dejar uno fuera, *leave one out*.

Por otro lado, se han hecho pruebas con otras señales de otra pieza. Estas señales se han mantenido fuera del conjunto de entrenamiento y han sido usadas sólo como *test*, para comprobar el rendimiento del sistema y comprobar la posibilidad de estar sobreentrenando el clasificador.

En la **figura 3-13** se representa una señal típica del conjunto A, con sus etiquetas de ground truth, el objetivo del clasificador en validación cruzada.

4.2.1 Resultados Back-end Baseline

El back-end baseline, basado en el máximo de las puntuaciones, da unos resultados bastante buenos, con un Fscore de 78 en validación cruzada. Sin embargo, es bastante deficiente en los *gaps*, ya que las puntuaciones del detector de gaps del back-end es las más ruidosa, como se puede ver en la **Figura Anexo-4-7,** obteniéndose un Fscore de 68 en *gaps*. Es nuestro objetivo mejorar esta puntuación.

4.2.2 Resultados en validación cruzada

En validación cruzada, sin usar ninguna transformación de variable, usando el back-end gaussiano multivariado estimado mediante ML, se obtienen buenos resultados, detectando todas las transiciones y las desviaciones. Sin embargo, se cometen errores clasificando pequeñas zonas de ruido como *gaps*, en la zona de desviaciones del final, como se puede comprobar en la **Figura 4-8** a continuación.

Se obtienen unos valores de *accuracy* del 95,27%, demostrándose que en este caso no es un evaluador de rendimiento válido; ya que, al tener unas clases muy desbalanceadas, una buena puntuación en el ruido (el evento más común) hace que el *accuracy* no represente bien el rendimiento del sistema. Sin embargo, se obtiene un *Fscore* de 81.74%, que sí que es más representativo de la realidad, siendo el *precision* del 73% y el *recall* del 92%. Este nivel bajo de *precision* se debe a las muestras que se etiquetan como evento cuando son ruido, sin embargo, el nivel alto de *recall* se debe a que todas las muestras de eventos son etiquetas como tal sin apenas fallo.

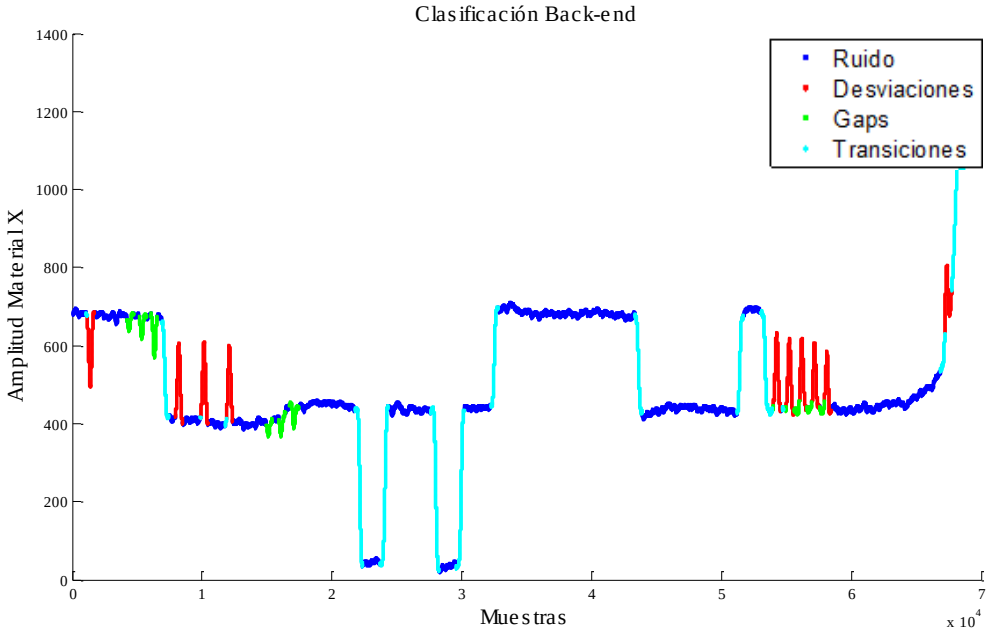

Figura 4-8: *Esta figura representa un ejemplo del resultado de clasificación en validación cruzada del back-end gaussiano entrenado mediante ML, sin transformación de datos. Se perciben errores de clasificación en las zonas de desviaciones.*

En la matriz de confusión de la **figura 4-9** se puede apreciar directamente este análisis. En la matriz de confusión los colores más claros indican porcentajes altos, y los más oscuros porcentajes bajos, como se puede ver en la barra de color de la derecha. Así pues, se puede identificar que hay algunas muestras de ruido que se clasifican como *gaps* y algunas muestras de desviaciones como *gaps* o ruido, aunque sólo son los bordes de los eventos, y carece de importancia.

Así, la matriz de confusión es útil para darse cuenta rápidamente de qué está fallando, mientras que el *Fscore* es una medida de rendimiento que sintetiza en un único número correctamente el rendimiento del sistema; comprobándose también que el *accuracy* no es válido para este caso debido al desbalanceo de las clases, al ser mucho más común el evento "ruido".

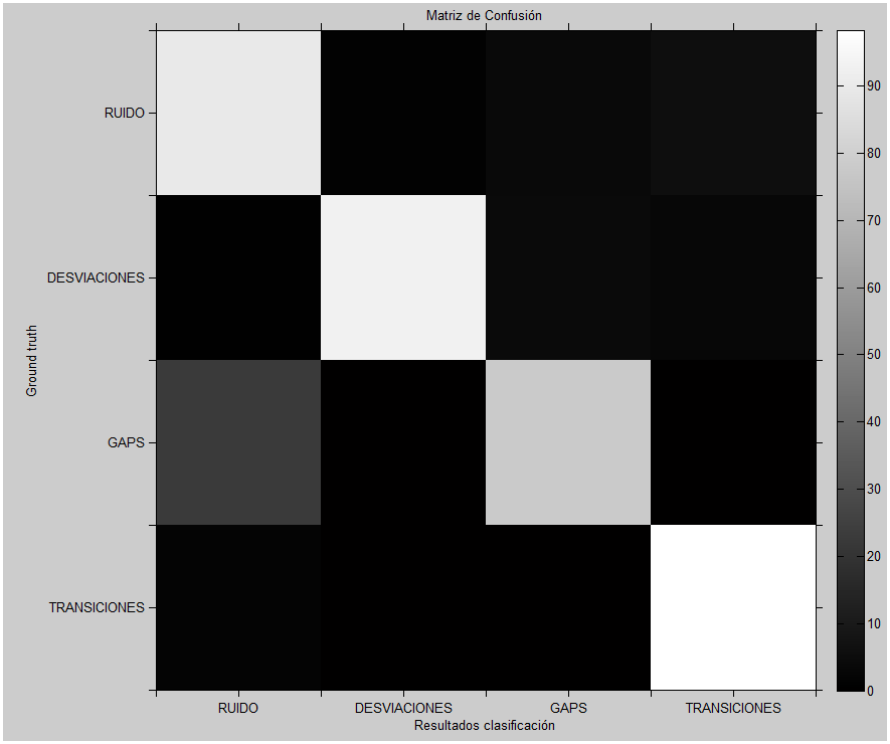

Figura 4-9: *Matriz de confusión de la clasificación. En cada cuadrado está en escala de color el porcentaje de muestras clasificadas respecto al total de muestras etiquetadas de ground truth de cada evento. Lo ideal es que estuviese el 100% en la diagonal, es decir, todos las muestras de cada eventos son clasificados como tal. Por ejemplo en los gaps, hay ciertas muestras que se han etiquetado como ruido aun siendo gaps, y por elo aparece en la columna de ruido un cuadrado de cierta luminosidad.*

4.2.2.1 *Usando transformación fija de variable*

En el caso de la transformación fija de variable, es decir usando el logaritmo de las puntuaciones, se mejoran estos resultados:

El *Fscore* aumenta hasta el 90.76, debido a que el *precision* mejora hasta el 92.84. La precisión del clasificador es mucho mejor debido a que las características están mejor modeladas por el back-end al usar el logaritmo.

El único fallo del clasificador es un gap que no se detecta correctamente y se etiqueta como ruido. Esto se puede ver en la matriz de confusión del Anexo A.

El cuadrado de la tercera fila, primera columna tiene un color claro, esto es porque ciertas muestras de evento gap han sido clasificadas como ruido también.

Estos resultados certifican la hipótesis de que las clases más separables y con forma más gaussiana, serían mejor modelas y tendrían un mejor rendimiento en el clasificador.

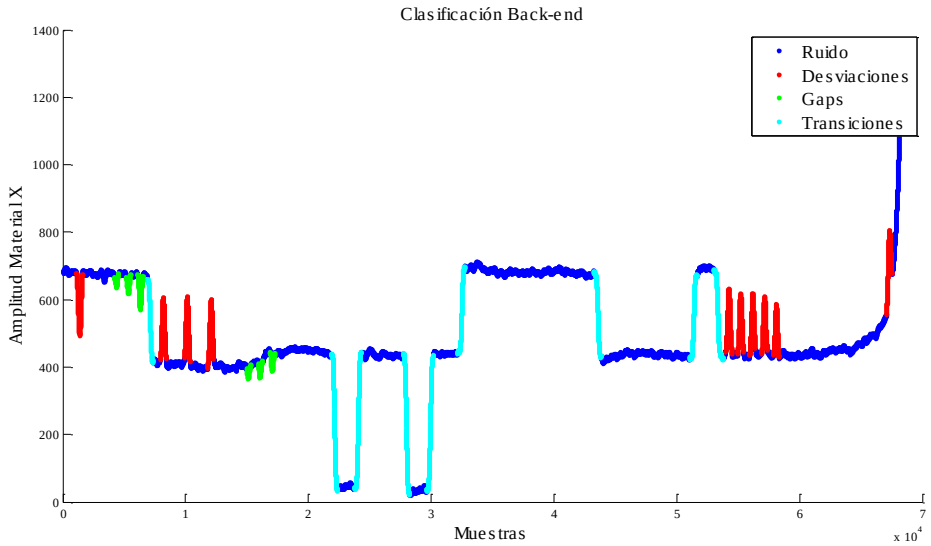

Figura 4-10: *Resultado clasificación tomando el logaritmo de las puntuaciones. Un gap es confundido con el ruido*

4.2.2.2 ***Con Feature Warping***

Se ha visto anteriormente que el feature warping, pese a conseguir un mejor modelado de los datos, provocaba que se perdiese capacidad de discriminación. Esto se comprueban en los resultados, que son muy malos, mezclándose las clases; consiguiéndose un *Fscore* de solo 64. El *accuracy* es del 94%, demostrando que no es un indicador útil en este escenario.

En la matriz de confusión se comprueban estos resultados, estando muchos eventos clasificados erróneamente como ruido.

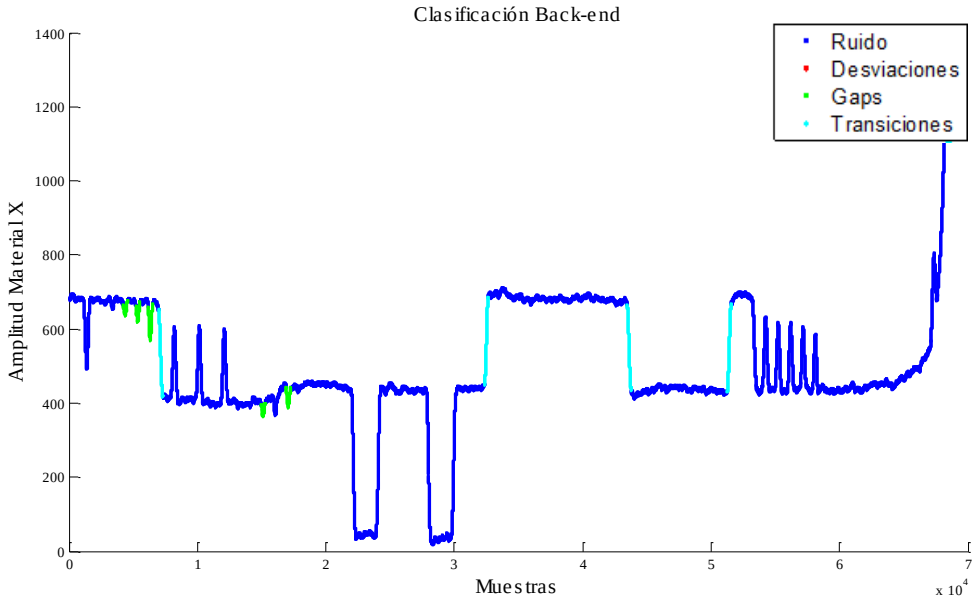

Figura 4-11: *Resultados de clasificación usando feature warping: La pérdida de discriminación hace que el back-end no sea capaz de clasificar correctamente los eventos*

4.2.2.3 *Diferentes tamaños de Ventana en el Front-end*

Cambiar ciertos parámetros del Front-end es interesante, como veremos más tarde, debido a que con las señales del grupo B, los eventos son de diferente tamaño como se verá en el punto **4.2.2**. Por lo tanto, a la hora enfrentarse a eventos más grandes es importante aumentar el tamaño de la ventana para que estos queden contenidos dentro y el Front-end pueda realizar una extracción de características correcta.

Al aumentar el tamaño de la ventana al doble que lo anterior, en el caso de la validación cruzada, con las señales del grupo A, se obtienen los resultados de la **Figura 4-12** usando la transformación con el logaritmo.

Al ser el tamaño de la ventana más grande, los eventos se unen entre sí y dejan colas más largas. Esto se refleja en el rendimiento como peor precisión, teniendo un *Fscore* del 73%. Es dudoso que en la realidad esta sea una peor clasificación, puesto que todos los eventos salvo el ruido son clasificados correctamente, con un 100% de *recall*, y no se clasifica como eventos zonas que no deberían serlo. El hecho de tener unas ventanas más grandes, hacen que los eventos se extiendan más a zonas de ruido.

De cara a mejorar la evaluación de rendimiento habría que tener esto en cuenta, como se hablará en trabajo futuro.

4.2.3 Resultados con señales de distinta procedencia

En este apartado se va a analizar los resultados del sistema con señales del grupo B, fuera del conjunto de entrenamiento, procedentes de la inspección de una pieza distinta, que no estuvieron disponibles hasta el final de este trabajo.

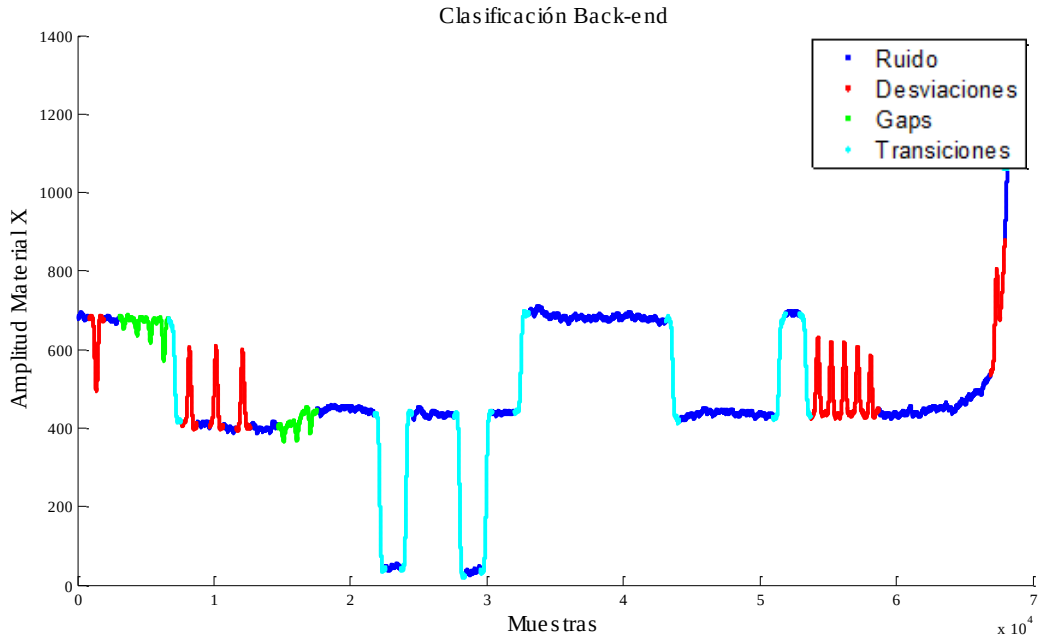

Figura 4-12: *La señal es clasificada correctamente, pero los eventos clasificados se extienden por el efecto de la ventana hacia el vecindario*

Se difiere con la del conjunto de entrenamiento en que carece de *gaps*, es decir, huecos en la pieza, y tiene desviaciones del material X mucho más pronunciados, además de un ruido mucho más potente.

Al ser los eventos más grandes en general, es necesario usar ventanas más grandes, para poder abarcarlos y poder realizar una buena extracción de características en el Front-end. Visto en el punto 4.2.1.3 que diferentes tamaños de ventana se pueden usar satisfactoriamente en el otro conjunto de señales analizadas mediante validación cruzada.

En la **figura 3-3**, están representadas las etiquetas de *ground truth* de la señal, se pueden ver las ocho desviaciones, de un nivel mucho mayor al de las barras de entrenamiento.

El sistema clasifica estas señales con un *Fscore* de 68, clasificando tanto las desviaciones altas y las zonas de ruido con mayor variación como *gaps*. El evento transiciones sin embargo consigue buenos resultados, como se puede comprobar en la **Figura 4-13.**

Estos resultados menos satisfactorios, son debidos a la gran diferencia que hay entre el grupo de *train* y el grupo de *test*, y es de esperar que añadiendo más diversidad de tipos de eventos al conjunto de *train* se consiga una mayor robustez en la clasificación.

4.2.4 Resultados Fusión

Existiendo dos Front-end distintos, uno funcionando mediante correlación y otro extrayendo características espectrales, existe la motivación de fusionar sus puntuaciones en el Back-end con el fin de alcanzar mejores resultados.

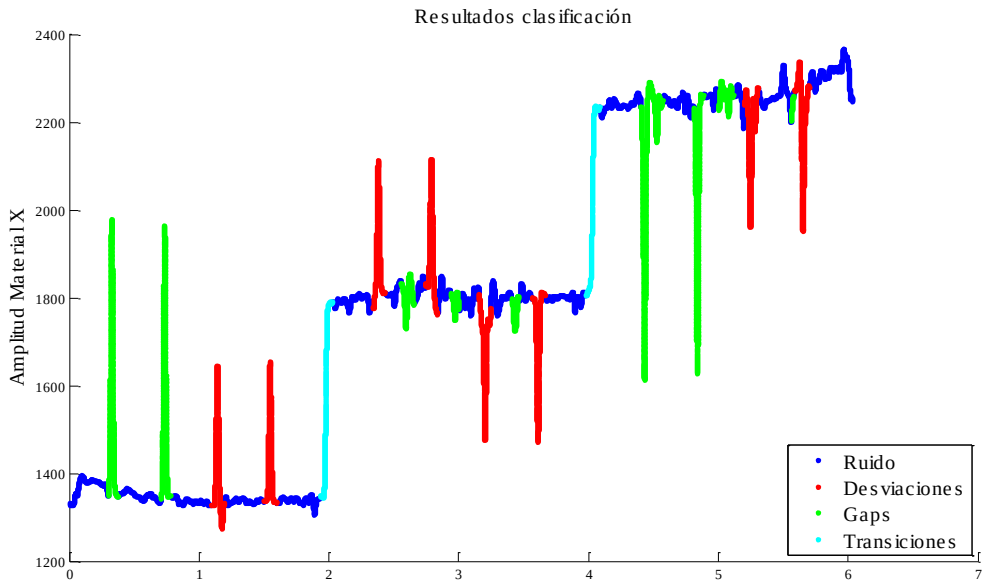

Figura 4-13: *El clasificador Gaussiano falla en los picos grandes, que clasifica como gaps, y también en zonas de ruido, que tiene más potencia que las señales de entrenamiento.*

Siendo los eventos de cada front-end distinto, es tarea del front-end la de mapear los eventos de cada front-end con los de la salida. Además, también es tarea del Back-end calibrar las puntuaciones de los dos detectores.

La mayor problemática de integrar los dos front-end es que mientras que el de extracción de características frecuenciales usa un esquema de enventanado, el de por correlación funciona muestra a muestra.

Con el fin de poder realizar la fusión de los dos grupos de características, se ha decidido usar un esquema de enventanado con una ventana deslizante en el Front-end por características espectrales. Por tanto, se evaluará la mejora de rendimiento del sistema fusión respecto a este esquema con ventana deslizante del Front-end por características espectrales.

Éste funciona prácticamente igual que el modelo base, como era de esperar, con un *Fscore* en cross-validation de 89.

En cross-validation, el esquema de fusión no mejora lo anterior, obteniéndose el mismo *Fscore*.

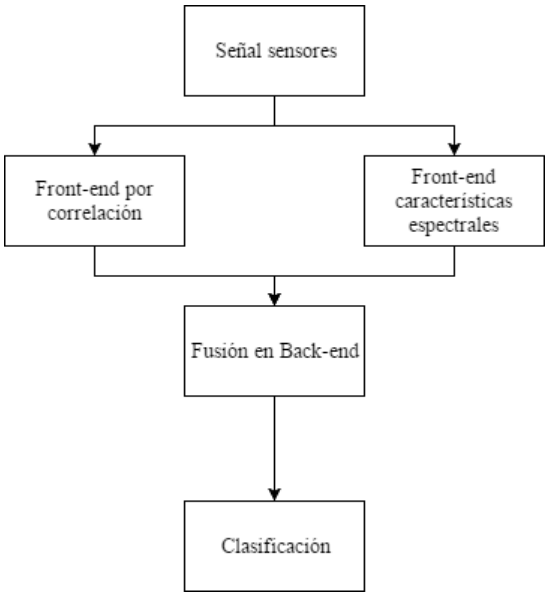

Figura 4-14: *Esquema del sistema usando fusión en el Back-end.*

Sin embargo, en el caso hacer el *test* con señales del grupo B, el resultado sí cambia, aunque no resulta de una mejora muy significativa; pasando de un *Fscore* de 68 usando solo el Front-end de características espectrales a 72 mediante la fusión con el Front-end de correlación.

Es de esperar que, añadiendo datos de entrenamiento más robustos, es decir más ejemplos de cada tipo de evento, se mejore el rendimiento del clasificador haciéndolo más robusto en otros escenarios.

4.3 Robustez Back-end estimado mediante modelo bayesiano frente a estimado mediante ML

4.3.1 Justificación

Una de las problemáticas existentes es la de la baja cantidad de datos de entrenamiento. En este escenario, es importante implementar modelos robustos a la escasez de datos.

Esta fue la motivación de estimar el Back-end gaussiano mediante un modelo bayesiano. Como se desarrolló en el apartado 3.1.3.4 el entrenamiento bayesiano da lugar a una T-student, que se caracteriza por tener unas colas más altas que la gaussiana cuando el número de datos de entrenamiento es bajo y es igual que la gaussiana cuando estos datos son muy altos.

4.3.2 Comparativa entre ML y Bayes con gran número de observaciones

Cuando el número de observaciones es alto, la gaussiana normal y la T-Student tienen un comportamiento idéntico. Todas las pruebas hechas anteriormente con el back-end gaussiano da resultados idénticos usando la T-Student, consiguiéndose la misma clasificación en cualquiera de los casos.

Al ser la gaussiana normal un caso particular en el cual el número de observaciones tiende a infinito, es de esperar que el caso general, es decir la T-Student, de los mismos cuando el número de datos de entrenamiento crece.

Es cuando el número de observaciones baja cuando se diferencian las distribuciones, y los resultados cambian. Por ello se ha realizado un *test* de robustez frente a la reducción de observaciones de uno de los eventos o de todos a la vez.

4.3.3 Robustez frente a la reducción de observaciones de un tipo de evento

En este apartado se va a evaluar la robustez del back-end gaussiano estimado mediante Maximum Likelihood frente al back-end gaussiano entrenado mediante Bayes, en el escenario de escasez de datos de entrenamiento de una de las clases.

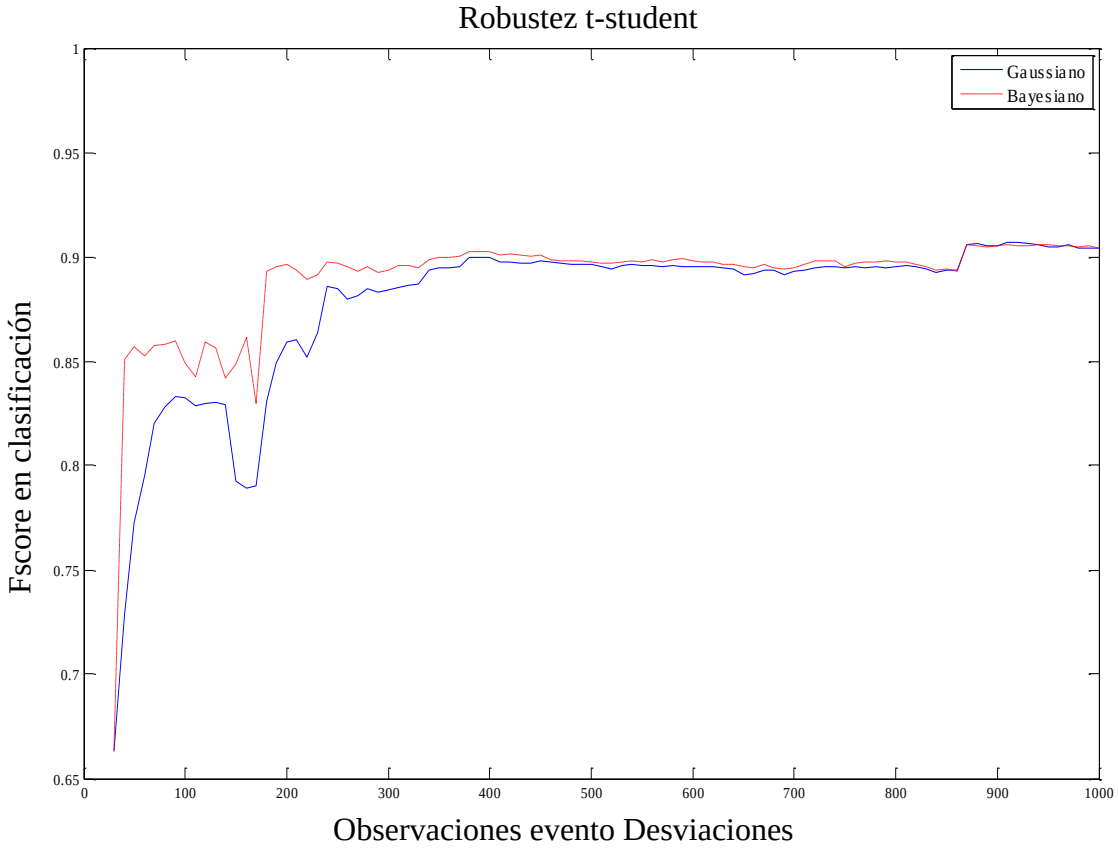

Figura 4-15: *Quitando observaciones de Desviaciones, el modelo Bayesiano tiene mayor robustez.*

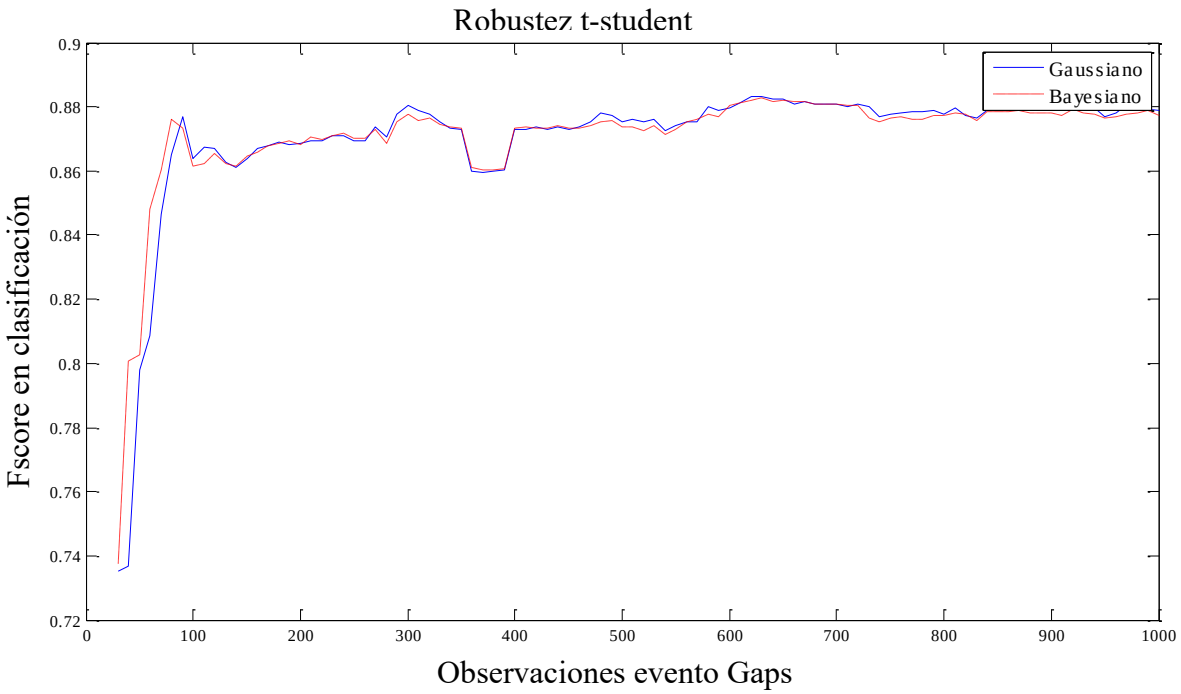

Figura 4-16: *En caso de quitar gaps, casi no hay diferencia entre los dos modelos.*

Cada una de las **figuras 4-15, 4-16, 4-17 y 4-18** representa el *Fscore* de la clasificación mientras se va aumentando el número de observaciones en entrenamiento de una de las clases aleatoriamente. Las demás clases permanecen constantes.

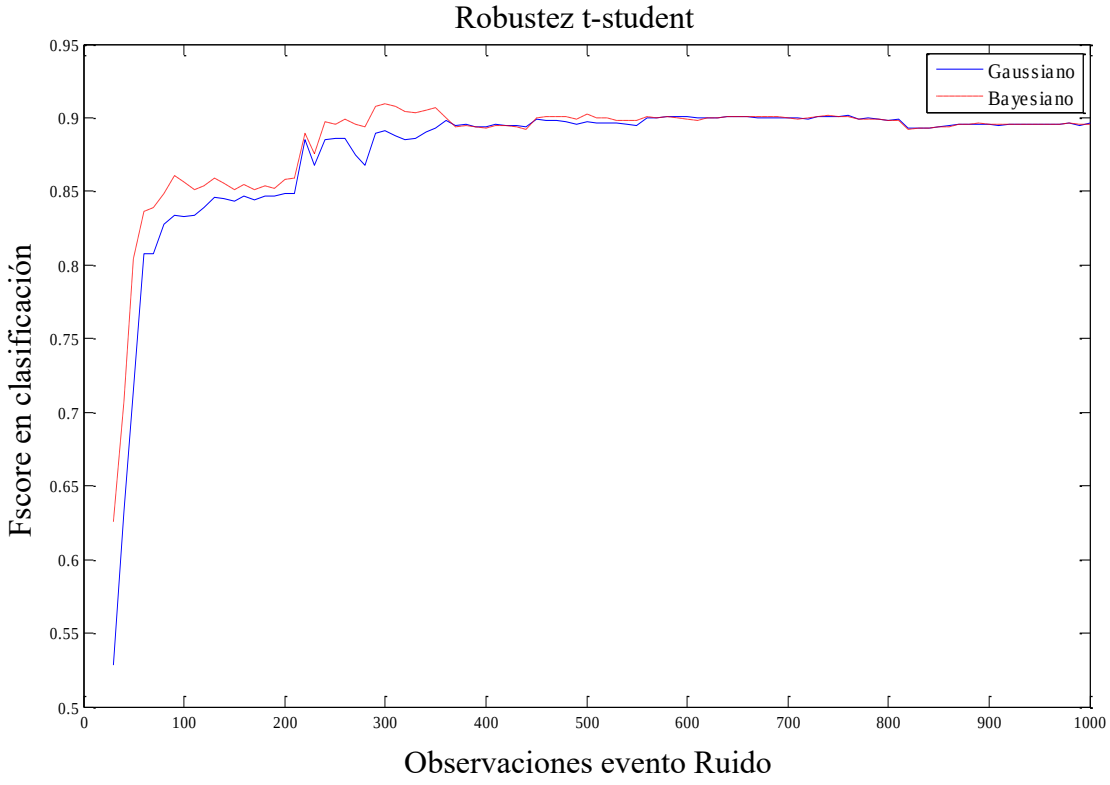

Figura 4-17: *En el caso del ruido el Bayesiano tiene una ligera ventaja al quitar muestras.*

En la figura 4-18 se puede comprobar, cómo al reducir el número de observaciones de un tipo de evento, en este caso las desviaciones en Material X, el back-end Bayesiano es más robusto, manteniendo un *Fscore* más alto. También es visible cómo cuando se aumentan los datos, el *Fscore* de los dos modelos es similar.

Con el evento *gaps*, se puede constatar el mismo efecto, aunque es menos claro que en el caso de las desviaciones.

En el caso del evento ruido, también existe esta ventaja en robustez del modelado Bayesiano frente al Gaussiano.

Y por último cuando se quitan observaciones de las transiciones también presentan una mejor respuesta del Back-end Bayesiano con respecto al Gaussiano Normal.

4.3.4 Robustez frente a reducción de observaciones de todos los eventos

En este caso, en el cual las observaciones de todos los eventos son reducidas y aumentadas poco a poco es donde se observa una mayor ventaja del back-end gaussiano con estimación bayesiana.

Estos resultados justifican plenamente el uso de este back-end frente al otro:

- En caso de un escenario con abundantes datos de entrenamiento, no habrá diferencia entre los dos resultados, siendo indiferente cuál usemos.

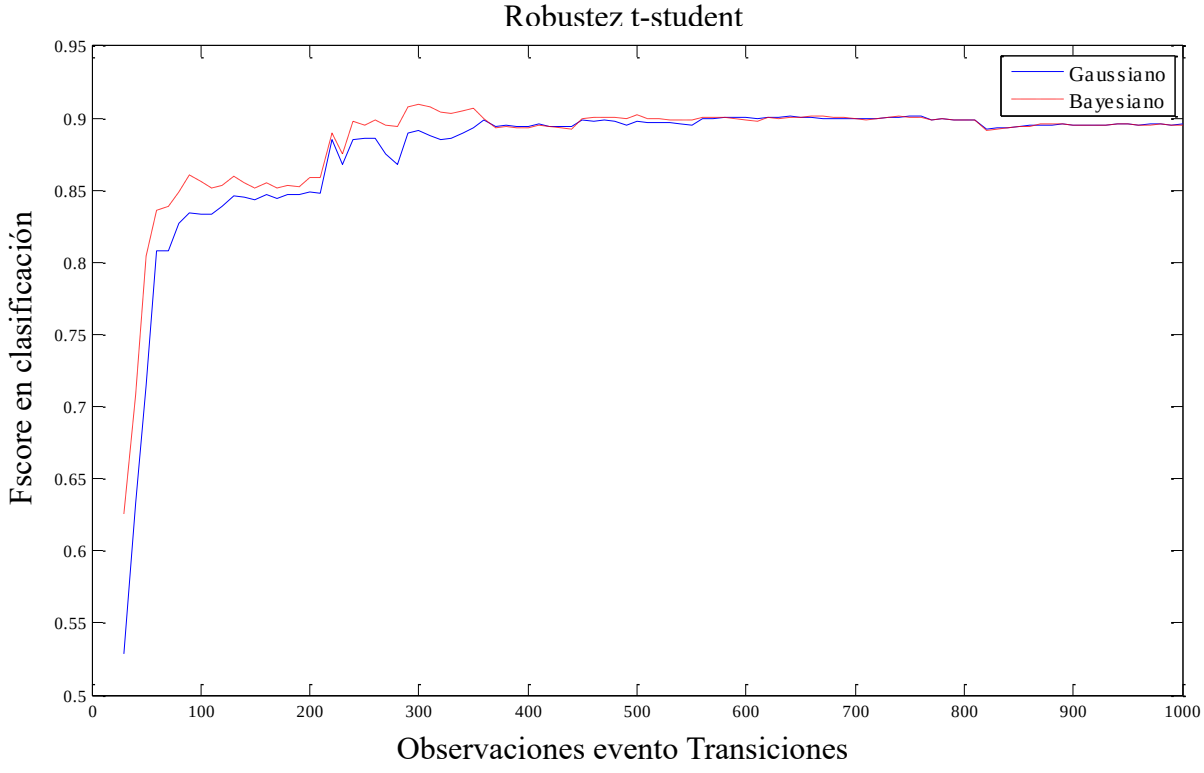

Figura 4-18: *Mismo comportamiento al eliminar transiciones*

Figura 4-19: *Al quitar todos los eventos a la vez, claramente la estimación Bayesiana aventaja a la estimación ML*

- Si existe carencia de datos de algún tipo de evento, o de datos de entrenamiento en general, la mayor robustez de la T-Student dará mejores resultados.

- La carga computacional es prácticamente la misma. El entrenamiento bayesiano puede parecer más costoso, pero en realidad, una vez integrados los marginales, sólo queda calcular la T-Student.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Se puede concluir, a partir de los resultados obtenidos y mostrados en la **tabla 5-1** que:

- El uso de back-end en este escenario está plenamente justificando, dando unos resultados robustos y coherentes.

- La estrategia de usar Feature Warping no ha sido la adecuada, ya que empeora los resultados al perder separabilidad entre los datos.

- Ha sido correcto realizar una transformación de los datos usando la función logaritmo, al ayudar en el modelado de los datos.

- El uso de una Gaussiana Multivariada como back-end ha sido una buena estrategia, mejorando los resultados del Back-end Baseline y permitiendo realizar la fusión de los Front-end correctamente y modelar bien los datos. Un back-end más complejo podría provocar overfitting.

- La estimación Bayesiana de la Gausiana Multivariada permite tener más robustez en un escenario de pocos datos frente a la estimación mediante Maximum Likelihood.

- La estrategia de fusión da mejores resultados y más robustos, al apoyarse en las fortalezas de los dos Front-ends.

Back -end	Fscore Validación Cruzada	Fscore Conjunto B
Base-line	78	32
Gaussiano sin Transformación de datos	81.7	61
Gaussiano + Logaritmo	90.7	68
Gaussiano + Feature Warping	64	43
Fusión	89	72

Tabla 5-1: Resultados globales en Fscore

5.2 Trabajo futuro

De cara al trabajo futuro en la clasificación de eventos en este escenario, se recomienda:

- Probar otros tipos de back-ends con la intención de mejorar el modelado de los datos.
- Usar como función objetivo en el Feature Warping otras distribuciones de probabilidad
- Desarrollar un sistema de evaluación de rendimiento más robusto, no basado en muestras si no la detección o no de los eventos en sí, lo cual es lo que se busca en la inspección de errores en piezas industriales.

6 Referencias

[1] Mikel Penagarikano, “Study of Different Backends in a State-Of-the-Art Language Recognition System”

[2] Haizhou Li, Bin Ma, Kong Aik Lee, “Spoken Language Recognition: From Fundamentals to Practice”

[3] Álvaro Iglesias Arias, “Extracción de Características para señales temporales procedentes de sensors industriales”

[4] Jason Pelecanos, Sridha Sridharan, “Feature Warping for Robust Speaker Verification”

[5] Duda, Richard O; Hart, Peter E; Stork, David G, “Pattern classification”

[6] Christopher Bishop, “Pattern Recognition and Machine Learning”, 2006

[7] Thomas P. Minka, “Inferring a Gaussian distribution”

[8] Michael Roth, “On the Multivariate t Distribution”

Anexos

A Figuras de interés

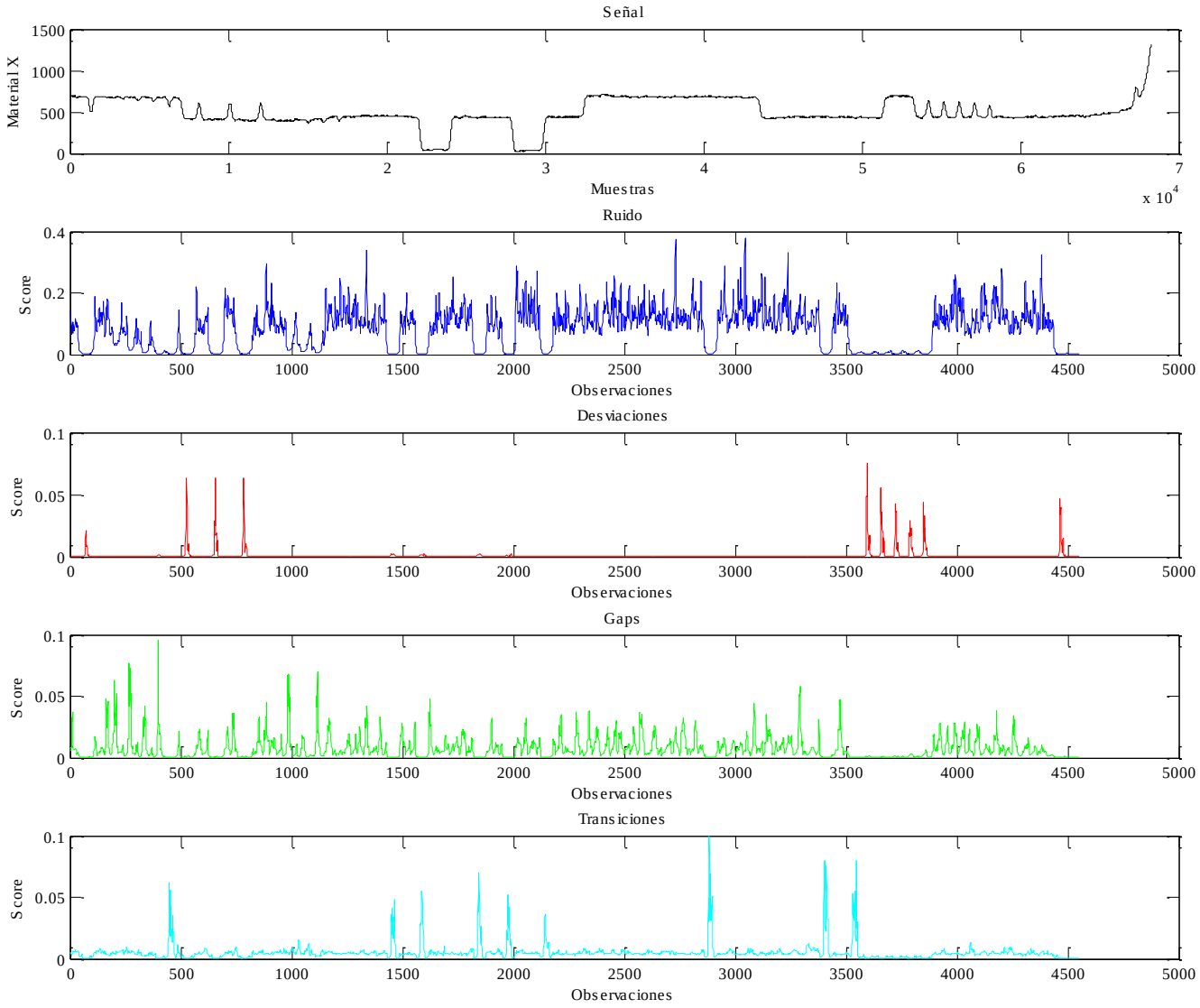

Figura Anexo-1: *Esta figura son los scores del front-end de características espectrales para una señal de material X dada. Como se puede comprobar, en las observaciones donde el evento en particular está presente la puntuación de ese evento del front-end aumenta.*

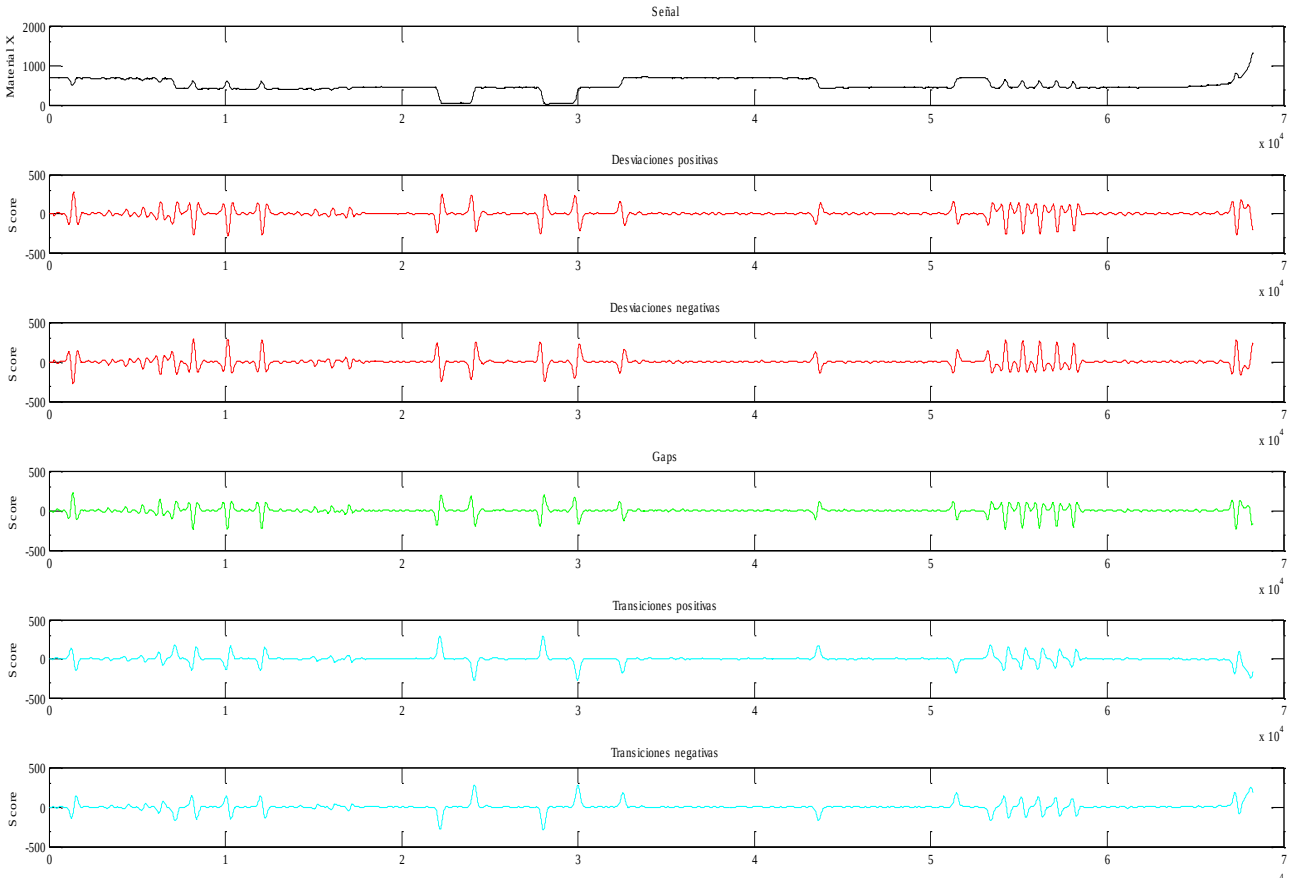

Figura Anexo-2: *Las puntuaciones del front-end por correlación reaccionan en los eventos. Por la naturaleza del front-end hay clases distintas que en el front-end por extracción de características, pero el back-end se encarga de mapear las clases correctamente, por ejemplo, las transiciones positivas y las negativas son mapeadas a la clase transiciones.*

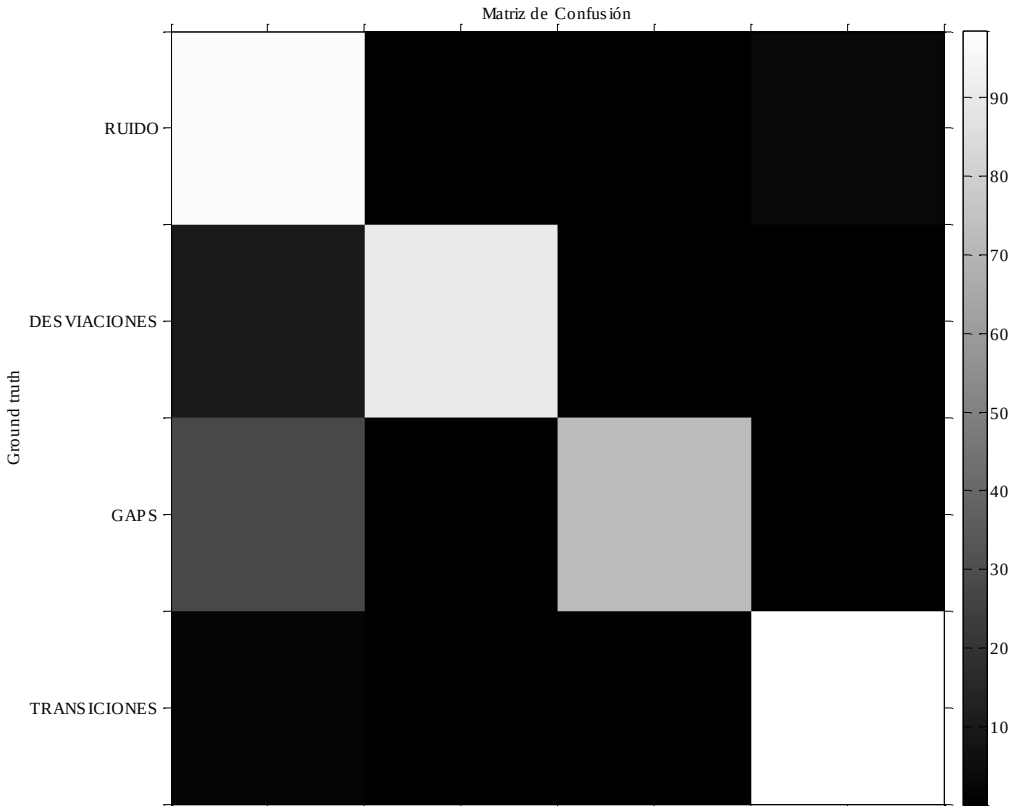

Figura Anexo-3: *Matriz de confusión usando logaritmo y Back-end Gaussiano estimado mediante ML.*